Multi-Model Heterogeneous Verification of Cyber-Physical Systems

Submitted in partial fulfillment of the requirements for
the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Akshay H. Rajhans

B.E., Electronics and Telecommunication Engineering, University of Pune
M.S., Electrical Engineering, University of Pennsylvania

Carnegie Mellon University
Pittsburgh, PA

May 2013

Keywords: cyber-physical systems, model-based design, heterogeneity, multi-model verification, behavioral semantics, semantic consistency

This thesis is dedicated to my loving family.

Abstract

Complex systems are designed using the model-based design paradigm in which mathematical models of systems are created and checked against specifications. Cyber-physical systems (CPS) are complex systems in which the physical environment is sensed and controlled by computational or cyber elements possibly distributed over communication networks. Various aspects of CPS design such as physical dynamics, software, control, and communication networking must interoperate correctly for correct functioning of the systems. Modeling formalisms, analysis techniques and tools for designing these different aspects have evolved independently, and remain dissimilar and disparate. There is no unifying formalism in which one can model all these aspects equally well. Therefore, model-based design of CPS must make use of a collection of models in several different formalisms and use respective analysis methods and tools together to ensure correct system design. To enable doing this in a formal manner, this thesis develops a framework for multi-model verification of cyber-physical systems based on behavioral semantics.

Heterogeneity arising from the different interacting aspects of CPS design must be addressed in order to enable system-level verification. In current practice, there is no principled approach that deals with this modeling heterogeneity within a formal framework. We develop behavioral semantics to address heterogeneity in a general yet formal manner. Our framework makes no assumptions about the specifics of any particular formalism, therefore it readily supports various formalisms, techniques and tools. Models can be analyzed independently in isolation, supporting separation of concerns. Mappings across heterogeneous semantic domains enable associations between analysis results. Interdependencies across different models and specifications can be formally represented as constraints over parameters and verification can be carried out in a semantically consistent manner. Composition of analysis results is supported both hierarchically across different levels of abstraction and structurally into interacting component models at a given level of abstraction. The theoretical concepts developed in the thesis are illustrated using a case study on the hierarchical heterogeneous verification of an automotive intersection collision avoidance system.

Acknowledgments

I thank my Ph.D. advisor Bruce Krogh for all his support throughout this Ph.D. Bruce has gone beyond the expected duties of an advisor so many times, from reading every sentence I have written in my papers and this thesis, to taking a mock qualifying exam after which the real qual was a breeze, to even holding one of my office hours when it was my quals week! His level of involvement has made remote advising from Rwanda work. I'd like to thank Prof. David Garlan, Prof. André Platzer, and Dr. Ken Butts for serving on my thesis committee and giving me valuable feedback during my Ph.D.

I owe a lot of gratitude to my other collaborators and co-authors. Clarence Agbi, Matthias Althoff, Ajinkya Bhave, Owen Cheng, Yi Deng, Alex Donzé, Agung Julius, Xin Li, Sarah Loos, Larry Pileggi, Ivan Ruchkin, Bradley Schmerl and Soner Yaldiz have co-authored various publications with me. Special thanks to Matthias and others for the work that won the 2011 ICCAD Best Paper Award and was later chosen as a research highlight by ACM. Collaboration with Prashant Ramachandra, Jim Kapinski and Jyotirmoy Deshmukh from Toyota has led to some interesting new research directions. Burt Andrews and Diego Benitez were my internship mentors at Bosch RTC and co-inventors on a patent.

My grad-student life has been pleasant due to the ~~window seat that Jim Weimer vacated for me~~ friends I've made. Ajinkya, Andrew, Aurora, Bosco, Dragana, Dusan, Ellery, the J's—Javad, Jim, João, Joel, Jon, Joya, Juhua, June and JY, Kshitiz, Kyle, Kyri, Le, Luca, Nick, Nikos, Matthias, Milos, Rohan, Sabina, Sanja, Sergio, Xiaoqi and Yilin have been fun to hang out with in and around Porter Hall. Luca, Sergio and Prof. Bruno Sinopoli deserve thanks for the nice coffee. Claire and Carol have kept things at ece-porterb running smoothly. Elaine and Samantha are excellent as Ph.D. program coordinators.

I made several friends while serving on the boards of Indian Graduate Students' Association (IGSA) and ECE Graduate Organization (EGO) at Carnegie Mellon and Maharashtra Mandal, Pittsburgh (MMPgh). Effie Barron at ECE helped with the EGO Insider's Guide updates. I had the pleasure of hosting a classical dance concert of the talented dancers and fellow grad students Anu, Brinda, Ishani and my wife Shriya. Pradeep Fulay has been a source of inspiration for endeavors both academic and otherwise. I owe my successful finish of the 2012 Pittsburgh Marathon to his motivation and encouragement.

My family back home in Pune has been a source of strength and support over all these years. Despite being 24+ hours of flight distance away, they are some of the closest people I have. Lastly, but most importantly, I thank my wife Shriya for helping me get through all the stress of grad school and keeping my sanity alive.

The work done in this thesis was supported by NSF grants CNS 1035800-NSF and CCF-0926181.

Contents

List of Figures

List of Tables

Notation

Symbol	Description	Page
M	model	20
$\mathcal{M}$	modeling formalism	20
S	specification	20
$\mathcal{S}$	specification formalism	20
B	behavior domain	20
$\mathcal{B}$	behavior class, behavior formalism	20
$[\![M]\!]^B$	semantic interpretation of model M in behavior domain B	21
$[\![S]\!]^B$	semantic interpretation of specification S in behavior domain B	21
$M \models^B S$	model M satisfies specification S in behavior domain B	21
$\Box\phi$, $\bigcirc\phi$, $\phi_1 \rightarrow \phi_2$	temporal logic operators 'always' ϕ, 'in the next step' ϕ and ϕ_1 'implies' ϕ_2	25
$R \subseteq B_0 \times B_1$	behavior relation R that associates pairs of behaviors (b_0, b_1) from $B_0 \times B_1$ that are related with each other	28
$R(B_0'), B_0' \subseteq B_0$	the subset of behaviors in B_1 that are associated with some behavior in B_0'	28
$R^{-1}(B_1'), B_1' \subseteq B_1$	the subset of behaviors in B_0 that are associated with some behavior in B_1'	28
Σ^*	the set of all finite-length traces from event labels in Σ	28
Σ^ω	the set of all infinite-length traces from event labels in Σ	28
$\mathcal{A} : B_0 \rightarrow B_1$	behavior abstraction function $\mathcal{A}$, a special case of behavior relation that is also a function	30
$C(P)$	constraint C on the set of parameters P	32
$\mathcal{C}$	constraint formalism	32
$[\![C^M(P^M), M]\!]^B$, or simply $[\![C^M, M]\!]^B$	semantic interpretation of parameterized model M in behavior domain B for all valuations of model parameters in set P^M determined by the constraint C^M	33

$[\![C^S(P^S), S]\!]^B$, or simply $[\![C^S, S]\!]^B$	semantic interpretation of parameterized specification S in behavior domain B for all valuations of specification parameters in set P^S determined by the constraint C^S	33
$C^M, M \models^B C^S, S$	parameterized model M with valuations of parameters P^M determined by constraint C^M satisfies parameterized specification S with valuations of parameters P^S determined by constraint C^S in behavior domain B	33
C_{aux}	auxiliary constraint C_{aux} capturing dependencies between various model and specification parameters	36
$P\|\|Q$	semantic composition of models P and Q	37
$b \downarrow^B_{B'}$, or simply $b\downarrow$	localization of behavior $b \in B$ to $b' \in B'$	40
$b' \uparrow^B_{B'}$, or simply $b'\uparrow$	globalization of behavior $b' \in B'$ to the set $\{b \in B \mid b\downarrow = b'\}$	41
P^G	globalization of model P	41
$P\|\|^G Q$	globalized semantic composition of models P and Q, same as $P^G\|\|Q^G$	42
$M_0 \sqsubseteq^B M_1$	model M_1 is an abstraction of model M_0 in behavior domain B	47
$M_0 \sqsubseteq^R M_1$	model M_1 is a heterogeneous abstraction of model M_0 via behavior relation R	49
$\mathcal{A}'\Uparrow$	globalization of behavior abstraction function $\mathcal{A}'$	59
$\mathcal{A}\Downarrow$	localization of behavior abstraction function $\mathcal{A}$	59
$S_1 \Rightarrow^B S_0$	specification S_1 implies specification S_0 in behavior domain B	73
$S_1 \Rightarrow^R S_0$	specification S_1 implies specification S_0 via behavior relation R	75
$\Box_{\mathcal{I}}\phi$	ϕ holds for the time interval $\mathcal{I}$ in immediate future	110
$\Diamond_{\mathcal{I}}\phi$	ϕ holds at some time during the time interval $\mathcal{I}$ in the immediate future	110

Chapter 1

Introduction

Cyber-physical systems (CPS) have tightly-coupled computational (cyber) and physical elements that have to interoperate in order for the systems to function. In today's world, CPS are everywhere: in transportation systems such as cars, planes and trains; healthcare and medical devices such as robotic surgery equipment and pacemakers; energy systems such as smart grids; smart infrastructures such as green energy buildings; industrial systems such as in chemical and nuclear plants; and robotics, to name a few.

Complex systems are designed using the *model-based design* (MBD) paradigm, where mathematical models of the systems are developed and checked against requirement specifications. The MBD approach aims to catch errors early in the design process before the system or prototypes are built, thereby avoiding costly re-design/re-development cycles [70]. MBD of CPS is a complex problem due to the tight coupling between the cyber and physical parts. The underlying theory, design principles and analysis methods for the many issues that need to be addressed in the development of CPS have evolved independently

in many distinct disciplines. Concerns that have been traditionally distinct, such as computation, timing, concurrency, algorithms, memory consumption from computer science and physical dynamics, stability, control, bandwidth limitations and transport delays from engineering need to be addressed together.

Heterogeneity is inherent in CPS because of the constituent physical dynamics, control logic, software implementation, real-time execution and communication networking. Therefore, for all but the most trivial CPS, many types of models need to be created and analyzed. There are several different modeling formalisms and analysis tools that suit different aspects of the overall CPS system design. Common formalisms and tools used for design and analysis of a CPS as depicted in Fig. 1 include: acausal equation-based models in tools such as MapleSim [3] and Modelica [4], suited for modeling the underlying physics of a system, e.g., the plant dynamics; signal-flow models in tools such as Simulink [6], suitable for control design and simulation; finite-state machines and labeled transition systems in tools such as LTSA [64], best suited for modeling decision logic and communication protocols; hybrid-dynamic models in tools such as SpaceEx[41], useful for analyzing abstract unified behaviors of continuous dynamics and discrete mode switches; network simulation models in tools such as OMNeT++[5], useful for analyzing communication network properties such as packet loss and communication delay; and software models in tools such as Spin [50], useful for analyzing whether the decision logic is correctly implemented.

There is no universal modeling formalism that can capture all possible design aspects for CPS. Therefore, MBD of CPS has to make use of not one, but a collection of different models in different formalisms. These models often make interdependent simplifying

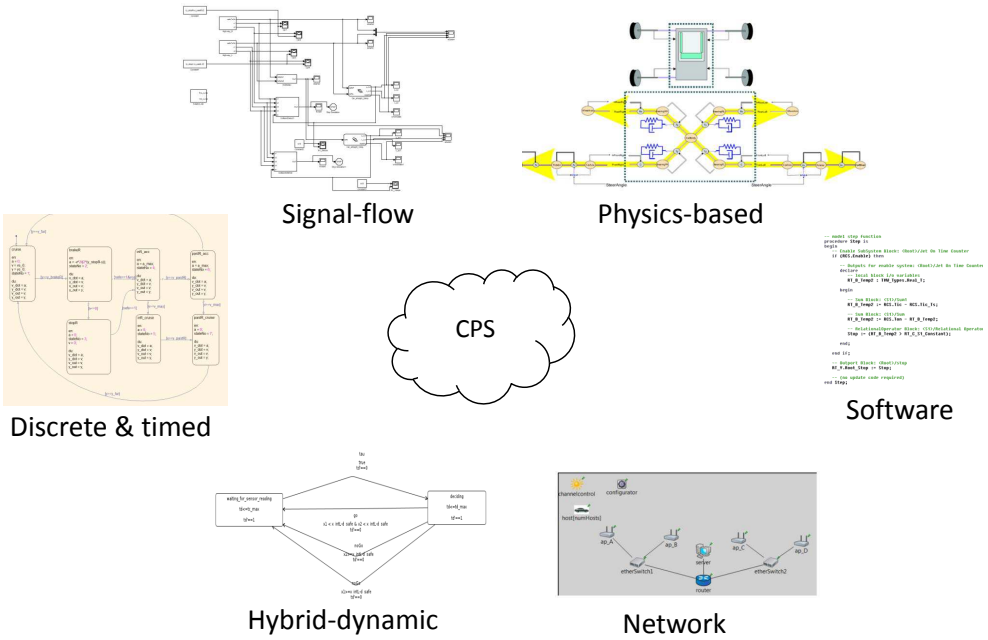


Figure 1.1: A typical collection of models and formalisms for CPS.

assumptions to occlude certain details while retaining those details that are relevant for a particular analysis at hand. Because of these dependencies, the models cannot be analyzed in isolation. This introduces the following problem of heterogeneity: without a single comprehensive modeling formalism, how can it be guaranteed that the heterogeneous models are consistent with each other, and how can verification results from the different formalisms be combined to infer system-level properties? In current practice, heterogeneity is addressed in an ad hoc manner. This thesis develops a formal approach to addressing heterogeneity in model-based design of CPS.

1.1 Heterogeneity in Model-Based Design

Support for heterogeneous formalisms remains ad hoc in current practice. Due to the lack of rigorous support, system designers often resort to extensive documentation in text files and spreadsheets about different models, specifications and simplifying assumptions. This approach is severely limited and lacks mathematical rigor, no matter how extensive the documentation. In the research community, several approaches from the research literature address different pieces of the overall heterogeneous model-based development puzzle; however, none presents a comprehensive solution. The research community has dealt with specific formalisms for specific purposes, but has not addressed heterogeneity at a general level without making specific assumptions about the formalisms involved.

Heterogeneous simulation tools, both commercially available ones, such as Simulink, along with its toolboxes and research tools, such as Ptolemy with its heterogeneous actor-oriented formalisms, aim to simply support simulation of heterogeneous aspects of systems together. Simulation is inherently informal and provides no formal correctness guarantees. There are several formal analysis approaches and tools that do provide formal guarantees but cannot be used in these simulation frameworks.

Other approaches to addressing heterogeneity involve model transformation using meta-models or model translation using interchange formats to convert one form of model into another. These approaches also have limitations. A lot of formalisms do not have exactly matching notions in their specific semantics, so often times one can only use a restrictive subset of modeling vocabulary that still works with model translation or transformation. Also, as research makes progress and new analysis tools and techniques become avail-

able, one has to re-create meta-models and translation schemes to be able to use the new formalisms necessary for the new tools and techniques.

Because of the lack of a general unifying framework for heterogeneous model-based development, the design and analysis approaches for CPS presently have several limitations. From a theoretical perspective, there is a lack of clear understanding about how commonly understood concepts for particular formalisms, such as abstraction, implication, composition, entailment can be generalized or extended to include multiple formalisms. As a result, the notion of verification of a system described using several heterogeneous models, how they relate to, or compose with, each other, and what their analyses mean in terms of the underlying system is not well understood. Due to the lack of clear understanding of how heterogeneous models and specifications relate with one another, there is no mechanism to combine analysis results together in a meaningful manner, except perhaps in approaches that treat analysis results as 'knowledge' or 'information' to be combined. Different aspects of system design are modeled in different formalisms and simplifying assumptions are inevitably made in creating those models. These assumptions are interdependencies in the semantic sense and there isn't a formal way of capturing them since they cut across formalism boundaries. Moreover, due to these interdependencies, models cannot be analyzed purely independently in isolation, but rather a mechanism to ensure these assumptions are consistent and that the analysis results are meaningful is necessary. In current practice, when models get analyzed in isolation, dependencies are captured informally at best, and there is no principled way of ensuring consistency. This thesis aims to address these limitations.

From a practical perspective, the lack of a unifying framework makes system integration difficult. Since there is no universal modeling formalism that can model all the aspects of the system design, creation of a system-level model in some modeling formalism isn't an option.

There is a need for a general formal framework that can support heterogeneous model-based development of CPS. The objective of this framework is to enable the *associations* between various heterogeneous models and their respective sets of behaviors in suitable behavior formalisms. The framework needs to be *general* enough to allow the use of different modeling and specification formalisms, and different analysis methods and tools. There needs to be *support for combining verification results* from individual models in a meaningful way to reason about the correctness of the overall system. Complex systems are usually composed of smaller subsystems where different parts are often designed by different engineers, possibly at different times. So the framework needs to *support compositionality and distributed development* in order to draw conclusions about the system correctness from the correctness of the components in a distributed compositional manner. Additionally, semantic assumptions and simplifications are made while constructing models from particular perspectives that hide or abstract information from other perspectives of the system. The framework needs the ability to *formally represent these assumptions* and *ensure system-wide semantic consistency* of these assumptions.

In this thesis, we develop this richer semantic support for heterogeneity to enable heterogeneous formal verification of CPS. Our heterogeneous model-based development framework lays out sufficient conditions and formal constructs that support the use of several

different system models together in a meaningful way. This thesis focuses on safety verification. The formal treatment ensures that different verification results can be combined with interdependencies resolved in a consistent manner; and the generality enables the use of many different modeling and specification formalisms and analysis tools together in a systematic manner.

1.2 Thesis Contributions

This thesis addresses some of the shortcomings of existing approaches outlined above. The contributions of this thesis are as follows.

1. *Behavioral semantics for heterogeneous models and specifications.* Behavioral semantics provides the generality necessary for supporting several different modeling and specification formalisms together in a common formal framework. Semantic interpretations of models and specifications can be defined in terms of subsets of behaviors from behavior domains of choice exhibited by the models or permitted by the specifications. This enables us to define the standard notions of abstraction between models, implication between specifications and entailment between a model and a specification in terms of subset of corresponding sets of behaviors within a given domain of their semantic interpretation.

2. *Semantic associations between heterogeneous formalisms using behavior relations.* Associations between different behavior domains provide a means to associate subsets of behaviors—particularly semantic interpretations of models and specifications—

from these domains. This enables the association between the semantic interpretations of models and specifications when they are not, or cannot, be defined in a common behavior domain.

3. *Abstractions across heterogeneous modeling and semantic formalisms using behavior relations.* Within our behavioral semantics framework, when semantics of two models are defined in a common behavior domain, the abstraction relation holds between the models when one overapproximates the set of behaviors of the other. We extend this notion of behavior set inclusion to allow heterogeneous behavior domains via behavior relations when the semantics of the two models are defined in different behavior domains.

   When multiple models are involved, we define a notion of behavior set coverage. Different subsets of behaviors of a given model are overapproximated by a collection of models such that every behavior of the underlying model is considered in at least one model. In this case, models individually are not abstractions of the underlying model, but together they form an abstraction. For mode-switching systems, we provide a notion of coverage with switching where behaviors of the systems in each mode can be covered by a different model. Such a coverage is useful when specifications require some temporally invariant condition to hold over all time, i.e., over all modes no matter how the system switches modes.

4. *Compositional approach to heterogeneous abstraction using behavior abstraction functions as special cases of behavior relations.* When component models are composed to form system models, heterogeneous abstraction can be performed independently

for each component so that one can infer abstraction for the composition from that of the component models. A framework based on behavior abstraction functions, a special case of behavior relations, is developed to elucidate sufficient conditions for compositional heterogeneous abstraction. Centralized and decentralized approaches to the compositional heterogeneous abstraction problem are considered and consistency conditions for the decentralized case are provided.

5. *Implications between heterogeneous specifications and their use for verification using behavior relations.* Formal specifiations are useful for expressing correctness requirements about system behaviors. Analogous to abstraction between models, we formulate implication between heterogeneous specifications based on the refinement of subsets of behavior allowed when the semantics of the two specifications can be defined in a common domain. For heterogeneous domains of semantic definitions, implication is extended using the associations provided by behavior relations, similar to model abstractions.

   In case of several specifications, a notion of conjunctive specification implication is developed whereby each specification can be focused on defining the behaviors of choice for a particular design aspect and its model while being allowed to be lenient on the others. Analogous to the notion of model coverage, conjunctive specification implication allows each specification to not have to individually refine the system-level specification, but when taken together they do.

6. *Hierarchical heterogeneous verification via behavior relations using nested conjunctive and disjunctive analysis constructs.* When using several models and specifications to-

gether towards system-level verification, we develop two types of analysis constructs. Using the notion of coverage that uses several models together for abstracting an underlying detailed model, we define a *disjunctive heterogeneous verification* construct wherein the individual models can be analyzed against specifications that are each stronger than the underlying system-level specification. Analogously, when models are each individual abstractions of the underlying system model, conjunctive specification implication can be used and verification of the models against these specifications can be used together in a *conjunctive heterogeneous verification*. Disjunctive and conjunctive verification constructs can be nested arbitrarily to form multi-level hierarchies of heterogeneous verification. A tree representation of hierarchical heterogeneous verification is presented for visualizing and managing conditions at each level within the overall system verification.

7. *Consistent heterogeneous verification in presence of semantic interdependencies.* During the creation of heterogeneous models of a system, simplifying assumptions are often made to facilitate separation of concerns. Details abstracted away in a particular formalism are usually captured in some other formalism. We use constraints over parameters as a uniform treatment across modeling and specification formalisms to formally represent these semantic interdepdenencies and develop parametric variants of hierarchical heterogeneous verification constructs. Auxiliary constraints model the interdependencies that cut across modeling and specification formalisms. Projections onto local sets of parameters for particular analysis tasks, given the interdependencies and the system-level parameter valuations to be considered, are used to capture

the external effect of the interdependencies on local verification tasks. Consistency conditions are developed for ensuring that these external effects of the interdependencies are captured in a correct manner when using interdependent analysis tasks together towards system verification.

8. *Demonstration of the practical applicability of the approach using a case study.* The theoretical machinery developed in this thesis is evaluated using a case study from automotive CPS domain. A cooperative intersection collision avoidance system is a heterogeneous safety-critical CPS that presents an ideal example to showcase our framework. We present a hierarchical heterogeneous verification tree for ensuring collision freedom for such systems. Different parts of the verification tree are used to demonstrate different aspects of the multi-model heterogeneous verification problem. We showcase how conjunctive and disjunctive heterogeneous verification constructs, coverage for mode switching, semantic consistency for parametric verification and compositional heterogeneous abstraction are all used towards a common system-level verification objective. All the previous theoretical contributions are illustrated using various elements of the heterogeneous verification case study.

1.3 Thesis Organization

We begin by outlining related work and summarizing its shortcomings in Chapter 2. Chapter 3 formulates the problem of heterogeneous model-based design in terms of models and specifications in various formalisms, their semantic interpretations in terms of sets of be-

haviors and semantic mappings using behavior relations. Chapter 4 develops the notion of abstraction between heterogeneous models using behavior relations, with coverage and compositionality conditions developed for using multiple models to establish abstraction. Chapter 5 introduces heterogeneous specifications for verifying the heterogeneous models and develops a hierarchical heterogeneous verification framework to enable the use of analyses of multiple models and specifications together towards system verification. Chapter 6 presents a hierarchical heterogeneous verification from the automotive CPS domain and showcases the different aspects of the theoretical machinery developed in this thesis used as different elements of the overall system verification activity. Chapter 7 summarizes the contributions of this thesis and outlines some future research directions.

Chapter 2

Related Work

There have been several efforts in the research literature in addressing different aspects of the overall heterogeneous model-based design problem for cyber-physical systems. This chapter reviews some relevant literature comparing and contrasting our approach from those, and mentions their shortcomings as a motivation for the work presented in this thesis.

Many efforts have focused on supporting simulation of heterogeneous elements or system models for multi-model system development. The field of computer automated multi-paradigm modeling (CAMPaM) is introduced in [68] and the current issues and a survey of promising approaches are outlined. Ptolemy II supports hierarchical integration of multiple "models of computation" into a single simulation model based on an actor-oriented formalism [20]. MILAN [61] is an integrated simulation framework that allows different components of a system to be built using different tools. The Metropolis toolchain [16] supports multiple analysis tools for design and simulation. SysWeaver [32] is a model-based

development tool that includes a flexible code generation scheme for distributed real-time systems. The functional aspects of the system are specified in Simulink and translated into a SysWeaver model to be enhanced with timing information, the target hardware model and its communication dependencies. The focus of these efforts has been simulation and not verification.

The Vanderbilt model-based prototyping toolchain provides an integrated framework for embedded control system design [77]. It provides support for multiple formalisms, such as functional Simulink/Stateflow models, software architecture, and hardware platform modeling along with deployment. The toolchain's ESMoL language has a time-triggered semantics, which restricts the functional view to Simulink blocks that can only execute periodically. There is currently no support for additional heterogeneity (e.g., due to physical or verification models), nor a notion of consistency between additional system models. Semantic anchoring to transform between system models transformation concentrates on the specification of the dynamic semantics of the domain-specific modeling languages [28]. The method relies on the observation that a broad category of component behaviors can be represented by a small set of basic behavioral abstractions such as finite state machines, timed automata and others. The underlying assumption is that the behavior of these abstractions is well understood and precisely defined. The methodology and toolchain is described in [27]. Unlike the semantic anchoring or toolchain approach for different yet specific formalisms, our treatment is general and works for any formalisms.

Several projects have focused on methods for transforming models between formalisms. Meta-modeling approaches such as Generic Modeling Environment (GME) [31], MILAN

[61], the Metropolis toolchain [16], and DEVS [86] enable heterogeneous model analysis by creating meta-models for each modeling formalism. The Hybrid Systems Interchange Format (HSIF) [71] and Automatic Integration of Reusable Embedded Software (AIRES) [46] use standardized interface formats to exchange information between multiple models. Translation schemes [28] and toolchains [77] have been developed to translate various types of models into these formats. The limitation of the model transformation approach is that with new formalisms and analysis tools that get introduced as research progresses, the translation schemes and meta-models need to be re-created before the new techniques can be supported. Additionally, in case of certain modeling formalisms and commercial tools, the semantics may not be precisely known to the system designer. In such a case, supporting those formalisms becomes difficult.

The work by Bhave et al. [21] provides a rigorous treatment of heterogeneity by using concepts from software architecture for disciplined engineering of complex systems. Through high-level, hierarchical component-oriented architectural models, their approach provides structural representations that form the basis for understanding the dependencies between various models that focus on partial analysis of the full system [80]. Structures of heterogeneous models are defined as views of the system structure from different design perspectives [22], and graph morphisms between system structures and model structures are used to ensure consistency across heterogeneous models in terms of correct deployment of functionality across subsystems [23]. While this body of work uses a rigorous treatment of heterogeneous models and multi-domain model-based development, it does not support system-level verification using heterogeneous models due to the lack of detailed operational

semantics at the architectural level.

The work by Julius [52] uses a behavioral approach in the spirit of Willems' work [87] and creates a framework for comparing and interconnecting behaviors based on the different time axis structures for discrete, continuous and hybrid behaviors. For embedded software applications, the Behavior-Interaction-Priority (BIP) framework [24] leverages the component structure of a system and supports behavioral annotation of the components by state diagrams to support system analysis [17]. In contrast to Julius's approach of incorporating behaviors in the definition of models, we see behaviors as the semantic interpretation of systems, which allows us to observe behaviors in different domains. This idea is similar to the one proposed in [47], where timed and time-abstract traces serve as different semantics for the same hybrid automaton. The notion of tagged signal semantics has been proposed to compare [62] and compose [19] heterogeneous reactive systems. Unlike [19, 24], the focus of this work is to use heterogeneous models independently towards a common system-verification goal, rather than to compose heterogeneous components into one big system for simulation.

Within the formal verification literature, the idea of using an abstraction in a simpler modeling formalism in order to verify safety properties about a more complex model in the original formalism has been used frequently. Hybrid abstractions of nonlinear systems [30, 48], linear hybrid automata abstractions of hybrid systems with linear dynamics [40], discrete abstractions of hybrid systems [10, 13, 29], continuous abstractions of hybrid systems [8] and synchronous abstractions of asynchronous modules [12] are some of the examples where simpler abstractions are successfully created and used. These approaches use

specific pairs of formalisms. Our objective is to create a general framework for abstraction that can support any set of heterogeneous formalisms.

NAOMI is an experimental platform for enabling multiple heterogeneous models to work together and keeping them and their data synchronized [34]. In NAOMI, a model is defined in terms of the set of input and output attributes that it shares with the system. NAOMI analyzes model dependencies to determine the impact of changes to one model on other dependent models and coordinates the propagation of necessary model changes. Inference-based approaches focused on ontologies have been proposed for static analysis and type checking [63]. In the similar spirit, the work in [59] focuses on integrating the results of disparate verification efforts and analysis techniques using static and epistemic ontologies. These approaches treat models, specifications, analysis results etc. as 'knowledge' and use the concepts from relational databases and targeted knowledge acquisition to put together 'knowledge' about the system. Rather than using ontology-based approach, we use a behavioral approach, which allows us to compare and relate behaviors of different types.

Analogous to our idea of logically nesting verification tasks, the TLA+ proof system deploys a proof manager that breaks down a complex verification task logically into proof obligations that are proved using theorem provers and satisfiability modulo theories solvers in case of software systems [26]. Our framework supports more general (e.g., continuous, hybrid) dynamics and non-deductive analysis methods. Verification architectures (VA) for real-time systems have been proposed to verify global properties by combining local analyses using abstract behavioral protocols in a deductive manner [37]. Unlike VA, whose aim is to use structured constructs like abstract protocols to simplify proof obligations,

our objective is to use structured constructs to put together system-level verification from the verification of heterogeneous models.

Compositional reasoning approaches in formal verification literature has focused on reducing the computational complexity of system-level verification by analyzing its constituent component models instead. Compositional methods, such as assume-guarantee reasoning, with abstraction defined by language inclusion [56] and simulation relations [38, 49], or compositional methods based on deduction [73], are usually defined in the context of a single formalism. Another example is the behavior-interaction-priority (BIP) framework for embedded software, which uses structured interaction invariants to support compositional analysis, but only for transition system models [18]. Our objective is to develop a general framework that elucidates the basic conditions for compositional abstraction between any pair of heterogeneous formalisms.

We summarize the shortcomings of the related work by noting that various pieces of the heterogeneity puzzle for model-based development of cyber-physical systems are addressed in specific contexts in the literature, but no approach provides a comprehensive general solution. In this thesis, we develop a general framework that within which any formalism can be used. Throughout the thesis, the addressing of the heterogeneity, the use of abstractions across different formalisms, formal verification using multiple models across various formalisms, compositionality and support for defining interdependencies across different formalisms and ensuring consistency, are all developed within a common framework.

Chapter 3

Behavioral Semantics for Heterogeneous Models and Specifications

The key challenge in successfully using heterogeneous models, specifications, analysis techniques and tools together to analyze an underlying system lies in being able to relate the different semantics of the various models and specifications. In this chapter, we formulate the problem of heterogeneous model-based design, define semantic mappings between heterogeneous formalisms and use them to define semantics of composition.

3.1 Problem Formulation

The objective of model-based design of systems is to create system models and analyze them against some correctness specifications in order to discover errors early in the development cycle. We begin by formally defining various elements of this process.

A *model* M is a mathematical description of a system using a *modeling formalism* $\mathcal{M}$, which is a collection of modeling primitives and syntactic rules for building models. Modeling formalisms typically used for CPS include transition systems, hybrid automata, signal-flow models, acausal equation-based models, and queuing networks. For simplicity of notation, we use the term modeling formalism synonymously with the set of all models that can be constructed using the primitives, and say a model M is an element of some formalism $\mathcal{M}$, written as a set membership $M \in \mathcal{M}$.

A *specification formalism* $\mathcal{S}$ is a collection of specification primitives for precisely defining correctness requirements. A *specification* S is a correctness requirement written in some $\mathcal{S}$. Various temporal logics, algebraic expressions that describe bad or unsafe sets of states to be avoided, are some of the specification formalisms used in capturing correctness specifications. We allow any such formalism, so long as the requirements can be written precisely and unambiguously. We also use the term specification formalism synonymously with the set of all specifications that can be constructed using the primitives, and say a specification S is a member of some formalism $\mathcal{S}$, written $S \in \mathcal{S}$.

The semantics of models and specifications is defined by a set of *legal behaviors* from a given *behavior domain* B in a *behavior class* $\mathcal{B}$. Behavior classes used to define semantics for CPS models include discrete traces, continuous trajectories and hybrid trajectories. For

each behavior class, we assume there exists a syntax, called the *behavior formalism*, which can be used to precisely define behavior domains and individual behaviors. For simplicity of notation, we use the terms behavior class and behavior formalism interchangeably, and use the set membership notation $B \in \mathcal{B}$ for a behavior domain B in the class/formalism $\mathcal{B}$.

For a model M in a modeling formalism $\mathcal{M}$, $[\![M]\!]^B$ denotes the set of legal behaviors in a behavior domain B from a behavior formalism $\mathcal{B}$ that it exhibits. Similarly, $[\![S]\!]^B$ denotes the set of behaviors that specification S written in a specification formalism $\mathcal{S}$ allows in a given behavior domain B from a behavior formalism $\mathcal{B}$. Note that $[\![M]\!]^B \subseteq B$ and $[\![S]\!]^B \subseteq B$.

In this thesis, we focus on the problem of *safety verification*, where the objective is to establish that model behaviors do not violate what is permitted by a given specification. When model and specification semantics are defined in a common behavior domain B, this safety verification requirement is mathematically captured as

$$[\![M]\!]^B \subseteq [\![S]\!]^B. \tag{3.1}$$

We say a model M satisfies a specification S in behavior domain B, written $M \models^B S$, if (3.1) holds. Fig. 3.1 shows a simple Venn diagram representation of safety verification problems using behavioral semantics. The behavior domain is shown as a gray rectangle. The semantic interpretations of model M and specification S are both subsets of the behavior domain B. Further, $M \models^B S$ if the set $[\![M]\!]^B$ is contained in $[\![S]\!]^B$. This ensures

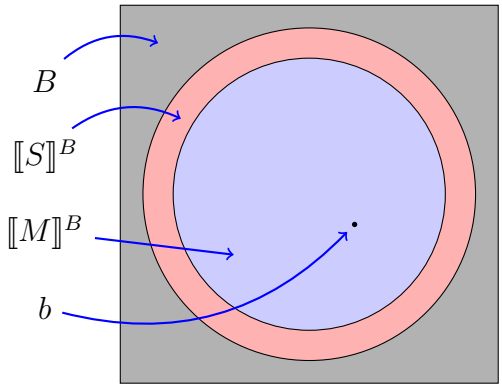


Figure 3.1: Safety verification problem in behavioral semantics.

that each individual behavior $b \in [\![M]\!]^B$ that the model M exhibits is allowed by the specification S.

Constructing a system or a model M such that it satisfies a specification S in a chosen behavior domain B is called a *design* task. On the other hand, checking whether a specification S is satisfied in a behavior domain B by a given system or model M is called a *verification* task. There can be several approaches to establishing $M \models^B S$ depending upon the particular formalisms. Examples of formal approaches for establishing safety verification include reachability analysis, theorem proving, establishing language inclusion using simulation relations, certificate-based guarantees [9], and numerical simulation with some formal guarantees such as sensitive state-space exploration [36] or robust testing [54]. Informal but principled approaches used in practice include exhaustive simulation with some coverage criteria [14, 55] and falsification [69, 83]. While we advocate that formal approaches should be used to establish verification whenever possible, we do not restrict what approach or tool is used to establish $M \models^B S$.

Suppose there were a universal modeling formalism $\mathcal{M}_0$ in which one could create an

all-inclusive model M_0 of a CPS. Suppose the system-level correctness specification S_0 is captured in a formalism $\mathcal{S}_0$, and the system behavior domain B_0 in a formalism $\mathcal{B}_0$ is used to define the semantics of M_0 and S_0. Then we would use some analysis technique to establish $M_0 \models^{B_0} S_0$ and we would be done. In reality, there is no universal modeling formalism $\mathcal{M}_0$ and even if there were, analyzing a gigantic all-inclusive system model M_0 would likely be intractable.

Fig. 3.2 shows the multi-model heterogeneous verification problem considered in this thesis. Rather than a single unified model M_0, several models M_i in different modeling formalisms $\mathcal{M}_i$ and their corresponding specifications S_i in specification formalisms $\mathcal{S}_i$ are usually created. The behavior domains B_i and behavior classes $\mathcal{B}_i$, as well as the tools used to establish $M_i \models^{B_i} S_i$ can be different for each i. Since the models M_i can have interdependent simplifying assumptions about other models as well as the underlying systems, these assumptions form interdependencies that cut across modeling and specification formalisms. The objective in this dissertation is to create a general framework to enable the use of heterogeneous models and tools to establish each of the entailments $M_i \models^{B_i} S_i$, and to combine these results to *conclude* $M_0 \models^{B_0} S_0$ without having to model or directly analyze M_0.

3.2 Models vs. Specifications

Models and specifications are both precise mathematical definitions of system behaviors; however, they usually differ in how they define the behavior set. Models are usually direct representations that typically give operational descriptions of the system dynamics, and

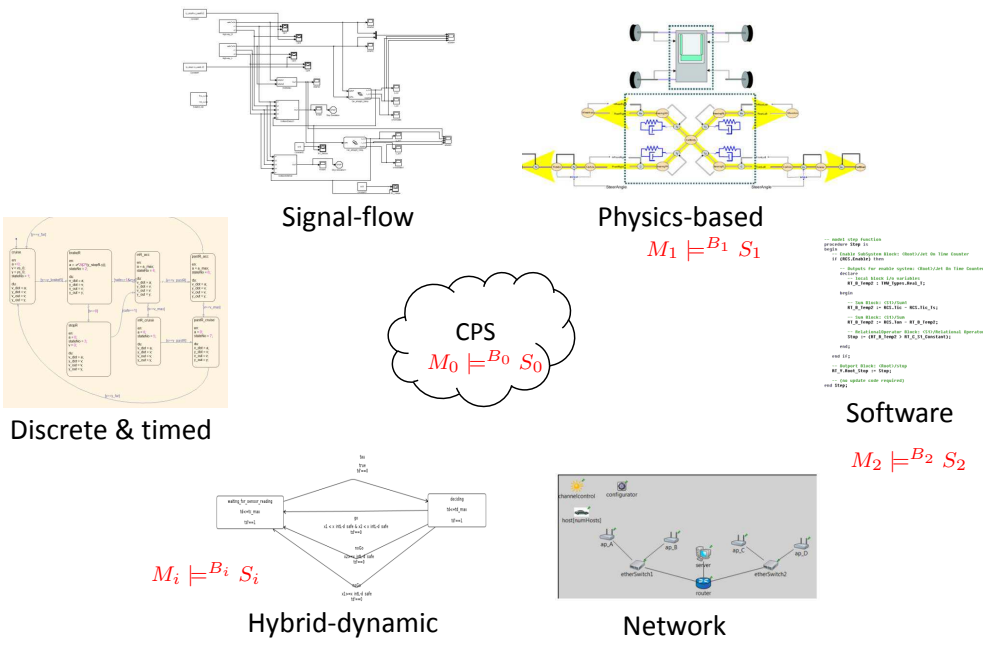


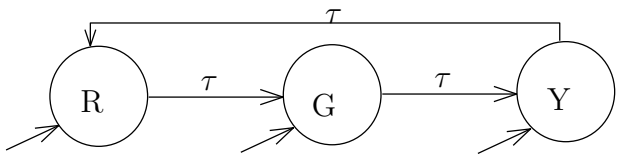
Figure 3.2: Verification using multiple heterogeneous models of CPS.

are therefore more suitable for developing implementations. They define *how* individual system behaviors or trajectories evolve over time by capturing the dynamics using, for example, differential equations, state transitions, or a combination of the two. Examples of such direct modeling formalisms include finite automata, transition systems, process algebras, differential equations, hybrid automata, hybrid programs, simulation models in Simulink and statecharts in Stateflow. Specifications, on the other hand, are usually indirect representations, which give declarative descriptions about sets of system behaviors and are therefore more suitable for capturing requirements. They describe qualitative information about the behaviors in terms of *what* properties they hold, irrespective of where the behaviors come from, or how they are generated. Various logics including several forms of temporal logic and mathematical expressions that define bad sets to be avoided are

popular formal indirect specification formalisms. We point out other terminology in the literature to essentially state the same divide. What we call models and specifications have also been called *operational* and *declarative* system descriptions [74], *system specifications* and *requirement specifications* [76], *constructive* and *axiomatic* style descriptions [60] and *requirement specifications* and *action specifications* [57].

Since models and specifications are both used to define sets of behaviors, semantically they are identical. It is because of this common semantic interpretation that we can use models as well as specifications in analogous semantic operations, sometimes even interchangeably. For example, the discrete interpretation of a traffic light can be represented in a transition system model as well as a temporal logic specification as illustrated in the following example.

Example 1. Consider a traffic system modeled using a transition system as illustrated below as model M. The system is allowed to start in any of the states, illustrated by the incoming arrows into the states with no origin, and transitions into the corresponding following states as per the allowed order $\cdots R \to G \to Y \to R \to \cdots$.



M

The same set of behaviors can also be defined using a temporal logic specification $S : \Box(R \to \bigcirc G) \wedge \Box(G \to \bigcirc Y) \wedge \Box(Y \to \bigcirc R)$, where predicates R, Y, G are true only in states R, Y, G respectively.

A sample behavior that both the model M exhibits and S permits is the trace b :

$G\ Y\ R\ G\ Y\ R \ldots$. It is easy to see that the sets of behaviors exhibited by M and permitted by S is identical. □

Under certain cases, models can be constructed from specifications. We call this process *modelization* of specifications. Linear-time temporal logic (LTL) admits a variety of efficient algorithms for translating a formula into an equivalent automaton [42, 43, 58, 84] and this construction is used in tools such as labeled transition system analyzer (LTSA) [64] and SPIN [50]. In case of real-time temporal logics, for metric interval temporal logic (MITL), timed automata construction has been proposed in [11] and [66]. In another work, timed automata and timed regular expressions have been proposed as interchangeable modeling and specification formalisms [15]. In the similar spirit, specifications can be reverse-engineered from models, a process called *specification mining* [51, 72]. Whenever it is not possible to find models or specifications that define exactly identical sets of behaviors, one typically finds the tightest underapproximation in case of modelization and the tightest overapproximation in case of mining. The particulars of these types of conversions are beyond the scope of this thesis.

3.3 Semantic Mappings Using Behavior Relations

The first step in analyzing heterogeneous models and specifications together in a common framework is to create a mechanism to compare their associated sets of behaviors. Typically, various modeling formalisms have their natural behavior formalisms that are the most suitable for defining model semantics. Examples include continuous trajectories

for ordinary differential equations, discrete traces for transition systems, hybrid trajectories for hybrid automata, and timed traces for timed automata. Analysis techniques and tools usually work with semantics developed in natural behavior formalisms for models in particular modeling formalisms. However, the semantics of models can also be defined in behavior formalisms other than the most natural ones. For example, the semantics of a hybrid automaton can be defined in terms of piecewise continuous trajectories ignoring the discrete transition labels, and in terms of discrete traces ignoring the changes to the continuous variables. Another example of semantics defined in different formalisms is when timed and time-abstract traces are used to define semantics of continuous and hybrid systems.

If semantics of various modeling formalisms can be defined in terms of a common behavior domain in a suitable formalism, that gives us a natural way of comparing and associating analysis results of each individual analysis task. In such a case, behavior domains B_i in behavior formalisms $\mathcal{B}_i$ would be identical for each i, which we can simply call $B \in \mathcal{B}$. This simple and straightforward approach is proposed in [78]. The semantics of a model can be defined by creating a mapping $\rho_{\mathcal{M}} : \mathcal{M} \to 2^B$, which for a model $M \in \mathcal{M}$ and given a common behavior domain B defines the corresponding set of all possible behaviors $\rho_{\mathcal{M}}(M) \subseteq B$ that the model exhibits. The semantic mappings for specifications are similarly defined by mappings $\rho_{\mathcal{S}} : \mathcal{S} \to 2^B$, which defines the behavior set $\rho_{\mathcal{S}}(S) \subseteq B$ that the specification S permits. The rationale behind creating such mappings to a universal behavior domain B is that models ultimately represent the same system, specifications ultimately restrict system behaviors and therefore, it should be possible to map model behaviors to system

behaviors irrespective of the particular modeling and specification formalisms.

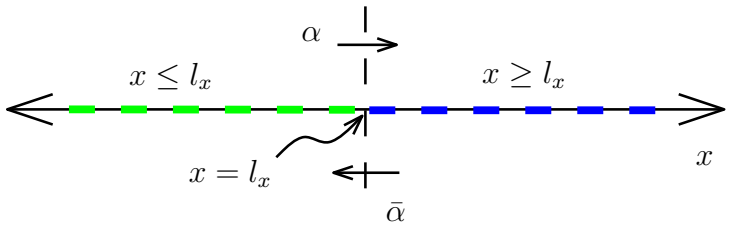
In general, when the semantics of models and specifications are defined using their natural behavior domains and there is either no obvious way of mapping them to a common behavior domain, or when using a common behavior domain makes analysis difficult, possibly because some tools that work in specific formalisms cannot be used in the formalism of the common domain, we need another mechanism to relate different semantics across heterogeneous behavior domains. We introduce *behavior relations* to deal with different behavioral domains. Behavior relations support semantic heterogeneity by allowing the use of several different types of behavior formalisms for different models and specifications [81].

Definition 3.1 (Behavior Relation)**.** Given behavior domains B_0 and B_1 in behavior formalisms $\mathcal{B}_0$ and $\mathcal{B}_1$, a *behavior relation* is a set $R \subseteq B_0 \times B_1$ that associates pairs of behaviors from the two behavior domains B_0 and B_1.

Given a behavior relation $R \subseteq B_0 \times B_1$, for a subset of behaviors $B_0' \subseteq B_0$, $R(B_0')$ denotes the set of behaviors in B_1 associated with behaviors in B_0', i.e., $R(B_0') = \{b_1 \mid \exists b_0 \in B_0' \text{ s.t. } (b_0, b_1) \in R\}$. Similarly, for $B_1' \subseteq B_1$, $R^{-1}(B_1')$ is defined as the set of behaviors in B_0 associated with behaviors in B_1', i.e., $R^{-1}(B_1') = \{b_0 \mid \exists b_1 \in B_1' \text{ s.t. } (b_0, b_1) \in R\}$.

Example 2. Consider a behavior domain $B_0 = \mathbb{R}^{\mathbb{R}_+}$ as the set of all 1-d continuous trajectories starting at time 0. Let the variable name for the single dimension be x, which we will call the *state*. Consider another behavior domain $B_1 = \Sigma^* \cup \Sigma^\omega$ defined as the set of all finite or infinite traces with event labels in $\Sigma = \{\alpha, \bar{\alpha}\}$. Consider a usual behavior abstraction technique frequently used in the literature — state-space partitioning,

illustrated below. The continuous state-space $\mathbb{R}$ is partitioned in two halves $x \leq l_x$ and $x \geq l_x$ at a boundary $x = l_x$ as follows.



The event corresponding to a continuous trajectory crossing the partition going from $x \leq l_x$ to $x \geq l_x$ is associated with the label α and that from $x \geq l_x$ to $x \leq l_x$ is associated with the label $\bar{\alpha}$.

We now express the semantic associations using a behavior relation R. Consider $b_0 \in B_0$ and $b_1 \in B_1$ where $b_0 = x(t)$ for $t \in \mathbb{R}_+$ and $b_1 = \sigma_0\sigma_1 \cdots$. In words, the behavior relation states that $(b_0, b_1) \in R$ if (i) $\exists$ a finite or infinite number of event times $t_i \in \mathbb{R}_+$, $i = 0, 1, \ldots$ that correspond to the continuous trajectory crossing the boundary in the right direction associated with the label σ_i (i.e., from "FROM(σ_i)" to "TO(σ_i)" according to the following table),(ii) there are no crossings between any consecutive event times t_i and t_{i+1} and (iii) if there is a final event time t_N, there are no crossings after that event time. Mathematically, these conditions can be written as

$$\forall t' \in [0, t_0), \qquad x(t') \in \text{FROM}(\sigma_0),$$

$$\forall t' \in [t_{i-1}, t_i), \qquad x(t') \in \text{TO}(\sigma_{i-1}) \cap \text{FROM}(\sigma_i),$$

$$\text{and if there is a final event time } t_N,\ \forall t' \geq t_N, \quad x(t') \in \text{TO}(\sigma_N),$$

where

σ	FROM(σ)	TO(σ)
α	$x \leq l_x$	$x \geq l_x$
$\bar{\alpha}$	$x \geq l_x$	$x \leq l_x$

Fig. 3.3 shows some continuous trajectories along the dimension x over time, overlaid on the two partitions of the state space. With respect to Fig. 3.3, for continuous behaviors c, d, e, f and discrete behaviors $g := \alpha$, $h := \alpha\bar{\alpha}\alpha\bar{\alpha}\alpha\bar{\alpha}\cdots$, $(c, g) \in R$, $(d, g) \in R$ and $(f, h) \in R$. In contrast, there does not exist any behavior b_1' s.t. $(e, b_1' \in R)$ since e never crosses the partition boundary.

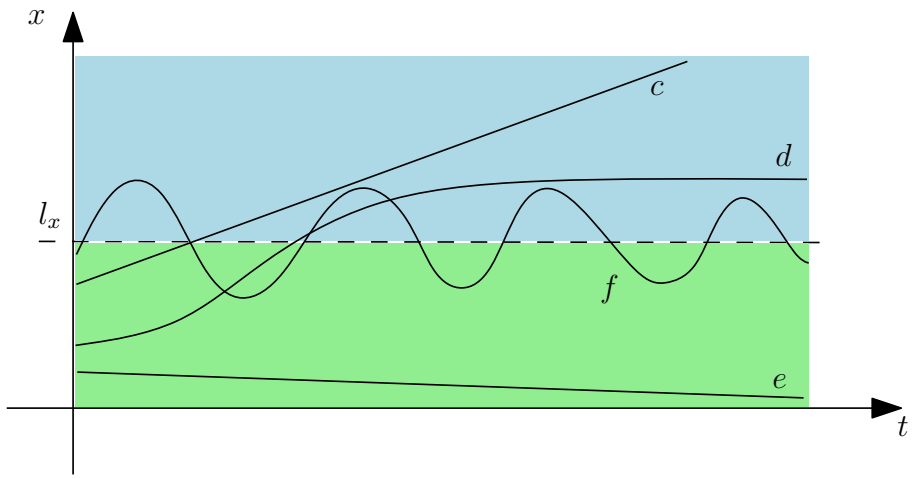


Figure 3.3: Continuous behaviors with state partitioning boundary.

□

We consider special cases of behavior relations that are also functions, i.e., $R \subseteq B_0 \times B_1$ s.t. $(b_0, b_1) \in R$ and $(b_0, b_1') \in R$ only if $b_1 = b_1'$. We call these *behavior abstraction functions*

and denote them as functions $\mathcal{A} : B_0 \rightarrow B_1$.

Example 3. We consider the behavior relation R from Ex. 2. It is already a partial function in that it does not associate any concrete behavior with more than one abstract behaviors, but not every concrete behavior has an associated abstract behavior. We can fix this by adding an association for all the remaining behaviors (i.e., those that do not cross the partition boundary) to empty behavior ε (which is an element of B_1).

With respect to Fig. 3.3, $\mathcal{A}(c) = \mathcal{A}(d) = \alpha$ and $\mathcal{A}(f)$ is the infinite string $\alpha\bar{\alpha}\alpha\bar{\alpha}\alpha\bar{\alpha}\cdots$, while $\mathcal{A}(e) = \varepsilon$. □

3.4 Addressing Semantic Interdependencies Using Parameter Constraints

Simplifying assumptions are made often in modeling different aspects of a system while creating different models in suitable formalisms. This approach supports separation of concerns where only those details that are relevant are represented, while the remaining details are abstracted away. These assumptions are based on details from models and specifications relevant to other aspects of the system. For example, as shown in Fig. 3.4, physical bounds on rates of change of variables, bounds on communication delays and computation times, error bounds on measurements capture pieces of information that belong to physical equation-based models, network models and timing models for worst-case execution time analysis, but can be used in a verification or control design model. To be able to formally model and address these semantic interdependencies in a manner

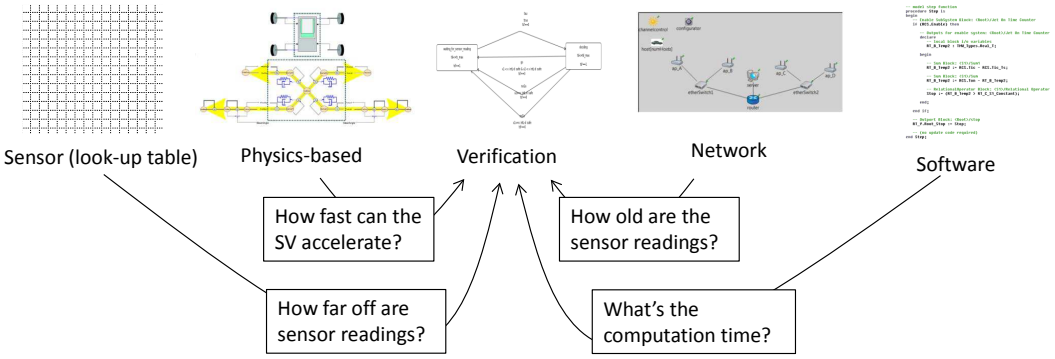


Figure 3.4: Interdependencies between heterogeneous models.

consistent across modeling and specification formalisms, we introduce constraints over parameters as a mechanism to make the interdependencies explicit. This approach has been presented in [78, 81].

A *parameter* p of a system is a real-valued variable that affects the system behavior. The *valuation* of a set of parameters P is a function $v : P \to \mathbb{R}$ that associates each parameter with a value. Note that parameters are required to be static, i.e., their valuation may be different for different behaviors, but it does not change over time. $V(P)$ denotes the set of all possible valuations of the parameters in P.

A *constraint* $C(P)$ over a set of parameters P is an expression written in a constraint formalism $\mathcal{C}$, such as first-order logic of real arithmetic. For a given $v \in V(P)$, $[\![C(P)]\!]_v \in \{\top, \bot\}$, where $\top$ and $\bot$ denote symbols for logical constants `true` and `false`, is the evaluation of the constraint $C(P)$ at v, and $[\![C(P)]\!]$ is the set of all valuations v over P for which $[\![C(P)]\!]_v = \top$.

Conjunction of constraints $C_1(P)$ and $C_2(P)$, written $C_1(P) \wedge C_2(P)$, is also a constraint whose corresponding parameter valuations are the intersection of the parameter valuations

of the original constraints, i.e., $[\![C_1(P) \wedge C_2(P)]\!] = [\![C_1(P)]\!] \cap [\![C_2(P)]\!]$. Similarly, disjunction of constraints is a constraint whose corresponding parameter valuations are the union of the parameter valuations of the original constraints. We write $C'(P) \Rightarrow C(P)$ when $[\![C'(P)]\!] \subseteq [\![C(P)]\!]$.

Given two sets of parameters P and P', the *projection* of a constraint $C(P)$ onto P', written as $C(P) \downarrow_{P'}$, is the constraint over P' defined by existential quantification of the parameters in $P \setminus P'$. Its valuations $[\![C(P) \downarrow_{P'}]\!]$ are

$$\{v' \in V(P') \mid \exists v \in [\![C(P)]\!] : v'(p') = v(p')\ \forall p' \in P' \cap P\}.$$

We now return to the problem formulation from Sec. 3.1. Fig. 3.5 shows the parametric multi-model heterogeneous problem. We let P_i be a set of parameters introduced for every i^{th} analysis task. These parameters can be in the models M_i or specifications S_i. Parameters $P_i^M \subseteq P_i$ are associated with the models M_i and parameters $P_i^S \subseteq P_i$ are associated with the specifications S_i. Constraints C_i^M and C_i^S determine the values of the parameters in P_i^M and P_i^S for models M_i and specifications S_i, respectively. The *semantic interpretation* of a parameterized model M_i with a constraint C_i^M, written $[\![C_i^M, M_i]\!]^{B_i}$, is the set of all possible behaviors in B_i associated with the model M_i for all parameter valuations in $[\![C_i^M(P_i^M)]\!]$. Similarly, the semantic interpretation of a parameterized specification $[\![C_i^S, S_i]\!]^{B_i}$ is the set of all behaviors in B_i that are permitted by S_i for the values of parameters P_i^S determined by the constraint C_i^S. The *parametric entailment* $C_i^M, M_i \models^{B_i} C_i^S, S_i$

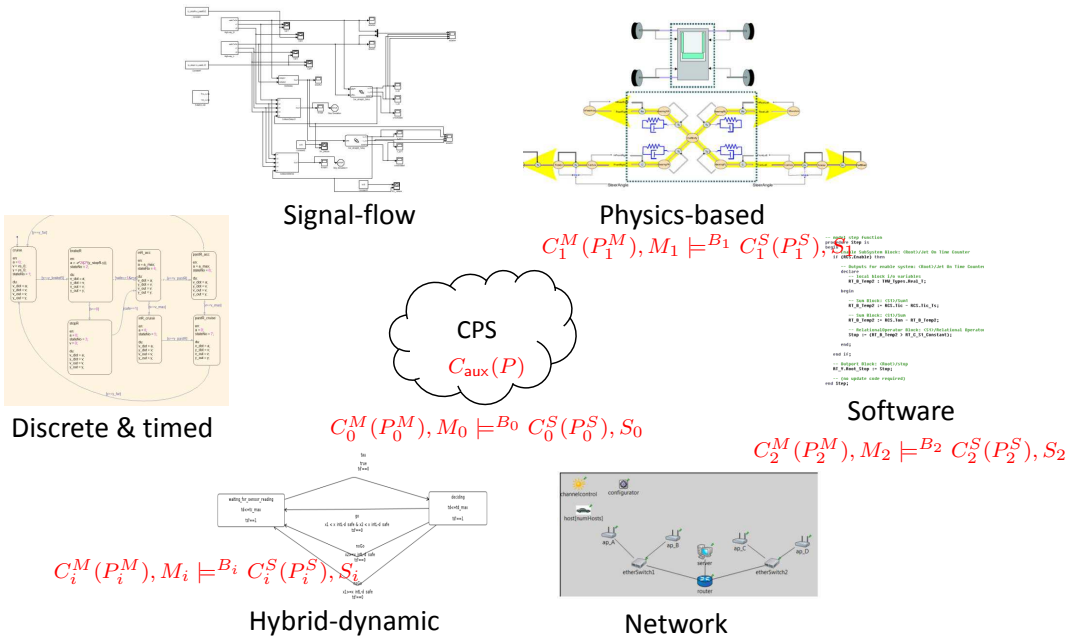


Figure 3.5: Parametric multi-model heterogeneous verification of CPS.

needs to establish that

$$[\![C_i^M, M_i]\!]^{B_i} \subseteq [\![C_i^S, S_i]\!]^{B_i}. \tag{3.2}$$

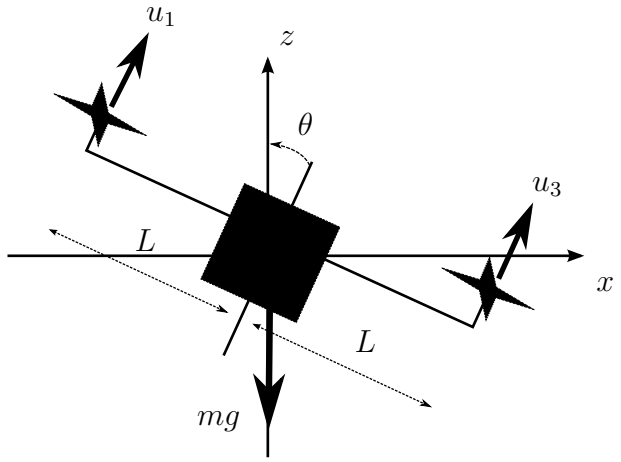
This equation is a parametric extension of Equation (3.1).

Example 4. We consider the STARMAC quadrotor model from [35]. The equations of motions for the quadrotor illustrated below are given by:

$$\begin{aligned}
\ddot{x} &= -\frac{b}{m}\dot{x} + \frac{1}{m}(u_1 + u_2 + u_3 + u_4)sin(\theta) \\
\ddot{z} &= -\frac{b}{m}\dot{z} + \frac{1}{m}(u_1 + u_2 + u_3 + u_4)cos(\theta) - g \\
\ddot{\theta} &= \frac{L}{I_y}(u_1 - u_3) - \frac{c}{L}\dot{\theta},
\end{aligned}$$

where x and z determine the position of the quadrotor in a two-dimensional vertical plane,

and θ represents its orientation with respect to the vertical axis. Here, the quadrotor mass m, constant c, length of the frame L, and inertia I_y are some of the parameters for the model and g represents the gravitational acceleration.



In absence of obstacles, the model uses an linear quadratic regulator (LQR) controller in a 'GoToTarget' mode to fly towards a target. In presence of obstacles, it switches to a 'GoUp' mode by giving equal inputs to all four motors. The safety specification is that the quadrotor always maintains a vertical safety distance v_{safe} from the ground, horizontal safety distance h_{safe} from potential obstacles in the 'GoUp' mode and velocity-dependent safety distance given by the velocity times some time-to-crash buffer t_{safe} from potential obstacles in the 'GoToTarget' mode. The parameters v_{safe}, h_{safe}, t_{safe} are specification parameters. The parametric entailment objective for this example is to establish that for all valuations of model and specification parameters of interest, the model satisfies the specification. Parameter values for which this parametric entailment holds, are synthesized in [35]. □

We observe that the set of possible behaviors of a given model grows or shrinks monotonically with increasing or decreasing sets of parameter valuations, i.e., if $C' \Rightarrow C$, then

$[\![C', M]\!]^B \subseteq [\![C, M]\!]^B$ for any model M. We assume that the specifications are parameterized such that increasing sets of parameter valuations allow increasing sets of behaviors, i.e., if $C' \Rightarrow C$, then $[\![C', S]\!]^B \subseteq [\![C, S]\!]^B$ for any specification S.

We let the constraint $C_{\mathsf{aux}}(P)$ denote the auxiliary constraints that capture the dependencies across the set of all parameters $P = \bigcup_{j=0}^{n} P_j$, which is the set of all parameters being used, including the original system-level parameters P_0. Without loss of generality we assume the sets P_j, $j = 0, 1, \ldots, n$ are disjoint.

Example 5. Suppose parameterized specifications S_1 and S_2 for a communication model and a computation model specify the communication delay and computation time as parameters τ_c and τ_{cc}. Suppose a verification model M_3 has a parameter ϵ that overapproximates the delays. This interdependency can be modeled using the auxiliary constraint

$$S_1.\tau_c + S_2.\tau_{cc} \leq M_3.\epsilon.$$

More complex nonlinear auxiliary constraints representing interdependencies appear in the case study in Sec. 6.6. □

The auxiliary constraint is qualified using the following requirement.

Definition 3.2. We say that an auxiliary constraint C_{aux} is *non-conflicting* for a given system-level constraint C_0 if

$$(C_0 \wedge C_{\mathsf{aux}}) \downarrow_{P_0} = C_0.$$

The definition states that the interdependencies themselves are not in conflict with the original system-level valuation of parameters for which we are to establish correctness

guarantees.

3.5 Semantics of Composition

Given the behavioral semantics of models defined as subsets of behaviors that the models allow, in this section we develop the semantics of composition of models. We start with a simple special-case scenario in which the semantics of component models P and Q are defined in the same behavior domain. In this case, we define the *semantic composition* of two component models as follows.

Definition 3.3 (Semantic Composition)**.** Given component models P and Q from the same modeling formalism $\mathcal{M}$ with semantics defined in behavior domain B, the composition $P||Q$ is a model in $\mathcal{M}$ s.t.

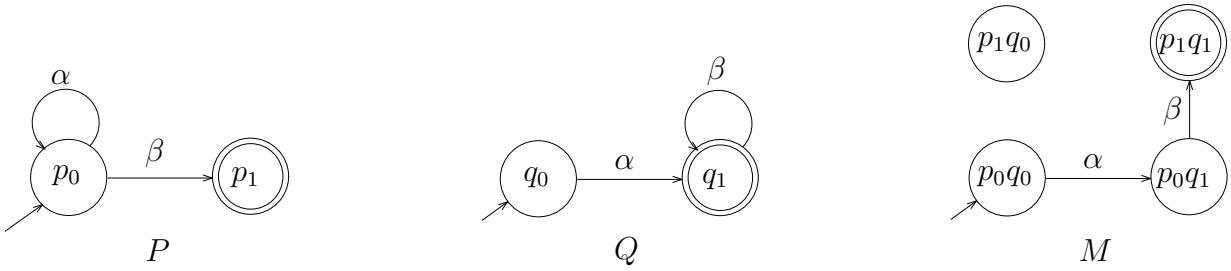
$$[\![P||Q]\!]^B = [\![P]\!]^B \cap [\![Q]\!]^B. \tag{3.3}$$

This definition of composition as the intersection of behavior sets is consistent with the literature for many definitions when the semantics are defined in a common behavior domain [19, 52, 62]. The definition, based on the system-theoretic work in the literature, e.g., by Willems [87], accommodates for the traditional concepts of component composition or interconnection using output-to-input connection. The traditional input-output definitions of component models do not specify the signal spaces for the inputs and outputs, which can be understood as being unrestricted. The output-to-input connection assigns equality constraint to the output of one component to the input of another, and the resultant behavior is one that agrees with both the components. For a detailed discussion, the reader

is referred to the work by Julius [52] that discusses how the interconnection of linear time-invariant systems, discrete-event systems and hybrid systems, as well as controller-plant interconnection can be defined using the notion of behavioral interconnection, defined as the intersection of corresponding sets of behaviors.

We note that for a given modeling formalism $\mathcal{M}$, syntactic techniques may exist for creating a composition, e.g., construction of product automata. We allow all such procedures to be used to define the composition, so long as (3.3) holds.

Example 6. Consider the following component models P and Q.



The behavior domain for the component models is $B := \{\alpha, \beta\}^*$, and the semantic interpretations are $[\![P]\!]^B = \{\alpha^*\beta\}$ and $[\![Q]\!]^B = \{\alpha\beta^*\}$. The semantic interpretation of a composite model $M := P||Q$ is $[\![M]\!]^B = \{\alpha\beta\}$, since $\alpha\beta$ is the only behavior that is common to the two[1]. Note that syntactic procedure of creating a product transition system results in the above composite system model M, whose semantic interpretation in B is exactly $\{\alpha\beta\}$. □

Example 7. Let $\dot{x} \in [1, 5]$ and $\dot{x} \in [2, 7]$ be the differential inclusion models of two components. Their composition is a model $\dot{x} \in [2, 5]$. Here the common behavior domain B for the two models is the set of all 1-d trajectories in the variable x. The physical intuition where this situation might arise is that say if x represents a variable in a physical

[1]Unlike some statechart semantics, such as Stateflow's, that allow unspecified events let a component model stay in the same state, in this example, the absence of a transition is blocking, meaning no action is possible.

process that evolves according to some physical constraints for each case, then for the combined case, physical constraints of both the cases have to be adhered to. In general, in compositions of models with differential dynamics, different components are allowed to control the evolution of different variables. We consider this case later in Sec. 3.5.3 where we define globalized semantic composition as the generalization of this special case that uses common semantic domains. □

The definition of semantic composition in Def. 3.3 can be extended to heterogeneous modeling formalisms if the semantics of the two models from two modeling formalisms can be defined in a common behavior domain B. An application for such a heterogeneous composition would be to mathematically compose controllers modeled in causal formalisms and plants modeled in acausal formalisms. Note, however, that for such a semantic definition of composition, there wouldn't be any syntactic constructs cutting across the two formalisms in order to give us the right set of composite behaviors.

Now we consider the semantic composition of component models when their natural behavior domains can be unequal. We create a mechanism to associate the component semantics to a common domain in which they can be composed. Mappings between this common domain and the natural domains of the component models are developed next.

3.5.1 Behavior Localization/Globalization

We now consider mappings between behavior domains from the same formalism. These sorts of mappings will be used to associate the semantics of a component model in terms of its local variables to the behavioral semantics at the global system level when it is

composed with other component models.

Definition 3.4 (Behavior Localization)**.** Given a behavior formalism $\mathcal{B}$ and two behavior domains $B, B' \in \mathcal{B}$, an onto function $\downarrow^B_{B'} : B \to B'$ (i.e., every element of B' has at least one pre-image in B) is called a *(behavior) localization* of behavior domain B to behavior domain B'.

Given a localization $\downarrow^B_{B'}$ of B to B', for $b \in B$, we will let $b{\downarrow^B_{B'}}$ denote $\downarrow^B_{B'}(b)$. In the following, we drop the superscript and subscript domain as indexes whenever the respective domains are clear from the context, or simply index them by the component names. Localization functions are extended to sets of behaviors in the usual way. Next, we consider two common types of variable elimination as examples of behavior localization.

Example 8 (Event label removal)**.** Consider behavior domains $B := \Sigma^*$ and $B' := \Sigma'^*$ for some $\Sigma' \subseteq \Sigma$. The localization of B onto B' is defined in terms of the event label removal for strings $s \in B$ as follows.

1. The empty string is projected onto itself, i.e. $\varepsilon\downarrow = \varepsilon$.
2. For any string $s \in B$ and event label $a \in \Sigma$
   - $(s \circ a)\downarrow = (s)\downarrow \circ\ a \ldots$ if $a \in \Sigma'$
   - $(s \circ a)\downarrow = (s)\downarrow \ldots$ if $a \notin \Sigma'$,

   where $\circ$ is a concatenation operator.

□

Example 9 (Continuous variable elimination)**.** Consider a global behavior domain $B = (\mathbb{R}^2)^{\mathbb{R}_+}$ of 2-d continuous trajectories, with the variables along the two dimensions named x and y. Let a local behavior domain be $B' = (\mathbb{R})^{\mathbb{R}_+}$ with the variable name x. Let

$\tilde{B} \subseteq B = \{[x(t)\ y(t)]^T |\ \forall t \in \mathbb{R}_+, x(t) \geq 0, y(t) \in [0,1]\}$. Then the localization due to elimination of variable y can be written in terms of its existential quantification. $\tilde{B}\downarrow$ can be defined as the set $\{x(t) | \exists y(t)$ s.t. $\forall t \in \mathbb{R}_+, x(t) \geq 0, y(t) \in [0,1]\}$. □

Definition 3.5 (Behavior Globalization)**.** Given behavior domains B, B' and a behavior localization $\downarrow$: $B \to B'$, for each behavior $b' \in B'$, the set-valued function $\uparrow : B' \to 2^B - \{\emptyset\}$ defined by $\uparrow(b') = \downarrow^{-1}(b') = \{b \in B | b\downarrow = b'\}$, is called a *(behavior) globalization* of B' to B.

For brevity of notation, we use $b'\uparrow$ to mean $\uparrow(b')$. Note that $b'\uparrow$ is always non-empty since the localization function $\downarrow$ is onto.

Behavior localization and globalization are generally inferred from relationships between models from given modeling formalisms and associated definitions of the relationships between model primitives and their semantic interpretations.

3.5.2 Model Localization/Globalization

Consider two component models P and Q with different local behavior domains B^P and B^Q in a behavior formalism $\mathcal{B}$. These local behavior domains have only the variables pertaining to a specific component while excluding others. In such a case, we need to lift the local semantics of the components to common global behavior domains before we can compose them.

We begin by defining the relationship between behaviors in a global domain and a local domain. Given the definitions of localization and globalization of behavior domains, we define model globalization as follows.

Definition 3.6 (Model Globalization)**.** Given a global behavior domain B, a model P with its local behavior domain B', and a behavior localization function $\downarrow : B \to B'$, the *(model) globalization* of P is any model P^G s.t. $[\![P^G]\!]^B = [\![P]\!]^{B'}\uparrow$.

For a given modeling formalism $\mathcal{M}$, syntactic approaches for globalization may exist, e.g., addition of self loops for newly added event labels for discrete transition systems, or addition of state variables with unconstrained dynamics for continuous dynamic systems. We allow the use of all such syntactic pre-processing procedures that lead to models with the correct set of behaviors $[\![P]\!]^{B'}\uparrow$ before composition.

Example 10. Consider two transition system models P as shown below. The local alphabet is $\Sigma^P = \{\alpha, \beta\}$ and the local behavior domain $B^P = \Sigma^{P^*}$. Let the global alphabet be $\Sigma = \{\alpha, \beta, \gamma\}$ and the global behavior domain $B = \Sigma^*$. Let the localization function $\downarrow : B \to B^P$ be defined as per Ex. 8.

The semantic interpretation of P in the local behavior domain B^P is the set $\{\alpha\beta\}$. The globalization of the set $\{\alpha\beta\}$ in the global behavior domain B is the set $\{\gamma^*\alpha\gamma^*\beta\gamma^*\}$. Note that the syntactic globalization procedure by introducing self loops for the new label γ results in a model P^G shown above, and $[\![P^G]\!]^B = [\![P]\!]^{B^P}\uparrow$. □

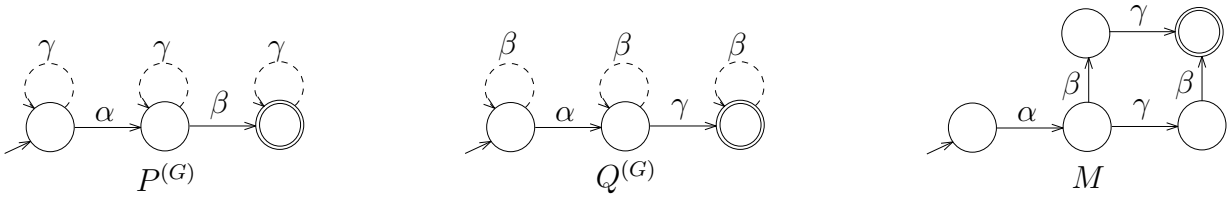
3.5.3 Globalized Semantic Composition

Given the above formal machinery in terms of model localization and globalization, the following definition generalizes the notion of semantic composition from Def. 3.3.

Definition 3.7 (Globalized Semantic Composition)**.** Given a global behavior domain B, component models P and Q with their corresponding local behavior domains B^P and B^Q, and behavior localizations $\downarrow^P : B \to B^P$ and $\downarrow^Q : B \to B^Q$, the *globalized semantic composition* of P and Q in the global behavior domain B, denoted by $P||^G Q$ is the semantic composition of models P^G and Q^G, which are the globalizations of P and Q respectively, i.e., $P||^G Q = P^G||Q^G$.

Syntactic procedures for globalization and composition can be used to construct globalized semantic compositions. Note that although syntactic procedures can produce different model globalizations, they still yield semantically equivalent compositions in terms of sets of behaviors. The non-uniqueness of syntactic globalization procedures is not an issue since the resulting globalizations are all semantically equivalent, which is what we need for the semantic composition.

Example 11. Consider two transition system models P and Q as shown below (without the dashed self loops).



$P^{(G)}$ $\quad$ $Q^{(G)}$ $\quad$ M

The local alphabets of P and Q are $\Sigma^P = \{\alpha, \beta\}$ and $\Sigma^Q = \{\alpha, \gamma\}$, and corresponding behavior domains $B^P = \Sigma^{P^*}$ and $B^Q = \Sigma^{Q^*}$ respectively. Let the global alphabet be $\Sigma = \{\alpha, \beta, \gamma\}$, and the global behavior domain $B = \Sigma^*$. Let $\downarrow^P$ and $\downarrow^Q$ be the localizations as defined in Ex. 8.

The local sets of behaviors for the two components are $[\![P]\!]^{B^P} = \{\alpha\beta\}$ and $[\![Q]\!]^{B^Q} = \{\alpha\gamma\}$. The semantic globalizations of the two component models yield $[\![P]\!]^{B^P}\uparrow^P = \{\gamma^*\alpha\gamma^*\beta\gamma^*\}$

and $[\![Q]\!]^{B^Q}\uparrow^Q = \{\beta^*\alpha\beta^*\gamma\beta^*\}$. The composition $M := P||^G Q$ has corresponding sets of behaviors given by $[\![M]\!]^B = [\![P]\!]^{B^P}\uparrow^P \cap [\![Q]\!]^{B^Q}\uparrow^Q = \{\alpha\beta\gamma, \alpha\gamma\beta\}$.

Note that the syntactic globalization procedure of introducing self loops yields models P^G and Q^G, whose syntactic composition results in a model M as shown above, s.t. $M = P||^G Q$. □

Example 12. Let $B^j := \mathbb{R}^{\mathbb{R}_+}$ be the sets of 1-d continuous trajectories with variable names x_j, $j = p, q$ respectively. Let two components P given by $\dot{x_p} \in [1, 2]$ and Q given by $\dot{x_q} \in [3, 5]$, respectively, have their semantics defined in domains B^j, $j = p, q$. Let $B := (\mathbb{R}^2)^{\mathbb{R}_+}$ be the system behavior domain of 2-d continuous trajectories with variable names along the two dimensions x_p and x_q. The globalizations of P and Q add the missing dimension and leave it unconstrained. Therefore, P^G and Q^G can be obtained as

$$P^G \equiv \begin{bmatrix} \dot{x_p} \\ \\ \dot{x_q} \end{bmatrix} \in \begin{bmatrix} [1,2] \\ \\ (-\infty, \infty) \end{bmatrix}, Q^G \equiv \begin{bmatrix} \dot{x_p} \\ \\ \dot{x_q} \end{bmatrix} \in \begin{bmatrix} (-\infty, \infty) \\ \\ [3,5] \end{bmatrix}.$$

Their composition is $P||^G Q \equiv \begin{bmatrix} \dot{x_p} \\ \\ \dot{x_q} \end{bmatrix} \in \begin{bmatrix} [1,2] \\ \\ [3,5] \end{bmatrix}$. □

3.6 Summary

In this chapter, we formulated the heterogeneous model-based design problem for analyses of the type system-level safety verification. Model semantics are defined in terms of their

interpretation in some behavior domain formally specified in some behavior formalism. Semantic mappings between behavior domains of different types are defined in terms of behavior relations and their special case behavior abstraction functions; while those between behavior domains of the same type are defined in terms of behavior localization and globalization.

The semantic mappings across heterogeneous behavior domains using behavior relations and behavior abstraction functions; and across homogeneous behavior domains using localization and globalization enable us to define model operations and relations such as heterogeneous associations and semantic composition. Given this theoretical machinery we study heterogeneous abstraction in the next chapter.

Chapter 4

Heterogeneous Abstraction

For all but the most trivial cyber-physical systems (CPS), abstraction is essential for making analysis and verification tractable. Different modeling formalisms are often used in various abstractions to facilitate the design of particular aspects of the system. In this chapter, we develop a formal machinery for establishing abstractions across any pairs of formalisms using behavioral semantics and behavior relations defined in the previous chapter.

4.1 Heterogeneous Abstraction Using Behavioral Semantics

Many different forms of abstraction have been considered in the literature. We focus on abstraction that corresponds to behavioral inclusion, for example, language or trace inclusion.

Definition 4.1 (Abstraction)**.** When semantically interpreted over the same behavior domain B, a model M_1 is an *abstraction* of a model M_0, written $M_0 \sqsubseteq^B M_1$, if

$$[\![M_0]\!]^B \subseteq [\![M_1]\!]^B. \tag{4.1}$$

This mathematical definition of a subset relation captures the notion of overapproximation, whose interpretation could depend on the particulars of the behavior formalism, e.g., in terms of language semantics, trace semantics and reachable sets.

In case of heterogeneous modeling formalisms, whenever it is possible to define semantics in terms of a common behavior domain, we can still use this standard definition of abstraction. Models M_0 and M_1 in the above definition can be from two different modeling formalisms $\mathcal{M}_0$ and $\mathcal{M}_1$. This is illustrated in the following example.

Example 13. Consider a Simulink model M_0 of a thermostat as shown in Fig. 4.1. The model depicts a simple illustration of heating and cooling dynamics of the temperature continuous variable, say T, when the heating is on and when the heating is off and the temperature drops towards the ambient. The hysteresis limits for switching the thermostat on and off, say $T_{\texttt{on}}$ and $T_{\texttt{off}}$, instantiated to values 14 and 16 units and the ambient and furnace temperatures T_a and T_f instantiated to -30 and 30 units are used for simulation. A particular initial condition ('initial value' of the 1/s block) is 10 units. In terms of the behavior domain B of piecewise continuous 1-d trajectories in the variable T, a simulation trace determines a corresponding behavior for the initial condition of choice, as shown in Fig. 4.2.

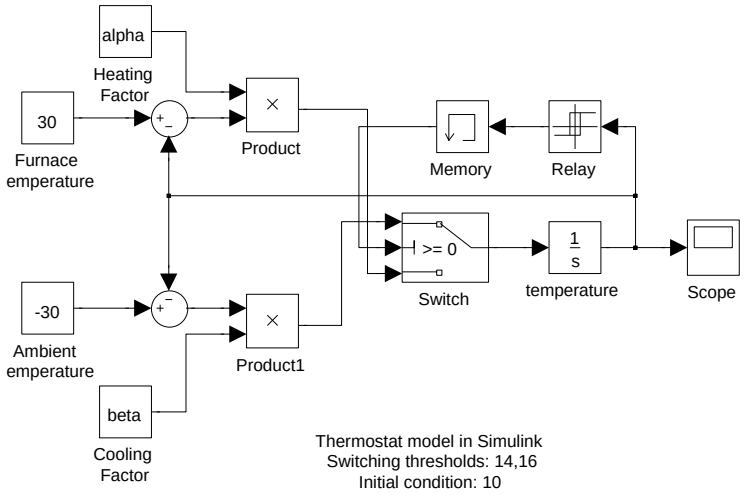


Figure 4.1: A Simulink model of a thermostat.

Now consider a hybrid automaton model M_1 of the thermostat as shown in Fig. 4.3. Let the semantics of M_1 also be defined in B. The behaviors of this model can be determined using reacheability analysis. If the ranges of the parameters $T_{\texttt{on}}$, $T_{\texttt{off}}$, T_a, T_f, and the bounds on the initial condition $\underline{T_0}$, $\overline{T_0}$ include the particular instantiations used in the Simulink model M_0, it can be shown that the hybrid automaton M_1 abstracts M_0 in B. The overapproximation of the corresponding behavior set $[\![M_0]\!]^B$ by $[\![M_1]\!]^B$ is given by the fact that the abstract automaton model allows more behaviors that those allowed by the Simulink model in domain B. □

The following definition extends the notion of behavior set inclusion from Def. 4.1 to heterogeneous behavior domains using mappings across the different domains to enable the subset comparison.

Definition 4.2 (Heterogeneous Abstraction)**.** Given behavior domains B_0, B_1 in behavior formalisms $\mathcal{B}_0$ and $\mathcal{B}_1$ and a behavior relation $R \subseteq B_0 \times B_1$, a model M_1 is a *heterogeneous*

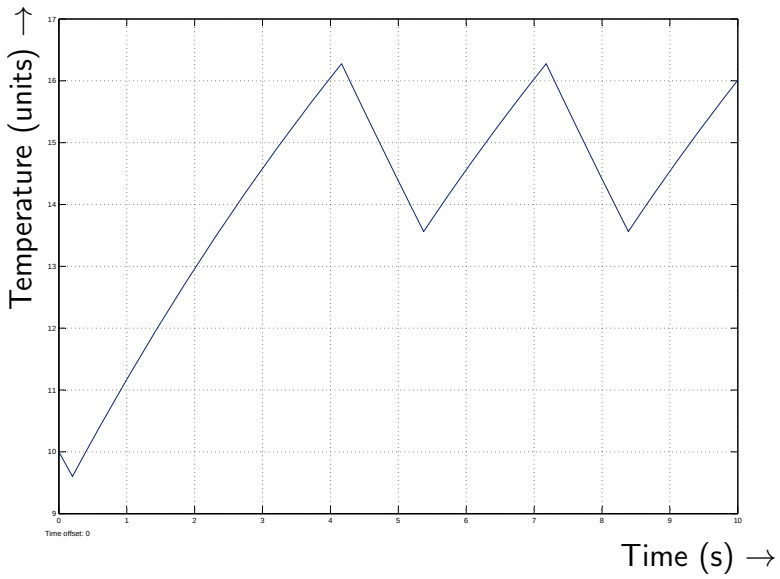


Figure 4.2: A behavior of the Simulink model.

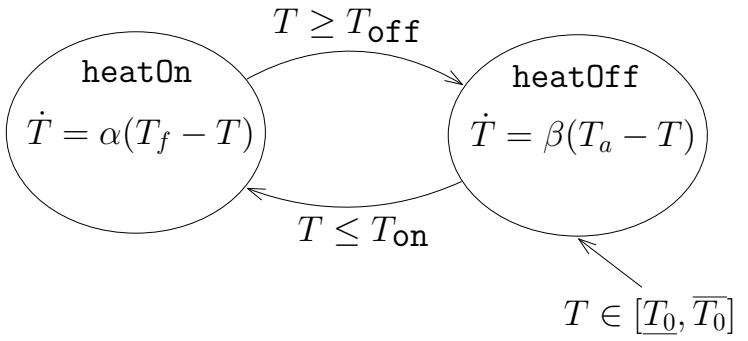


Figure 4.3: A hybrid automaton model of a thermostat.

abstraction of a model M_0 through R, written $M_0 \sqsubseteq^R M_1$, if

$$[\![M_0]\!]^{B_0} \subseteq R^{-1}([\![M_1]\!]^{B_1}). \tag{4.2}$$

Fig. 4.4 show a Venn diagram representation for the heterogeneous abstraction definition using behavioral semantics. It asserts that for every behavior b_0 in B_0 of model M_0, the behavior relation R associates at least one corresponding behavior in B_1 of model M_1.

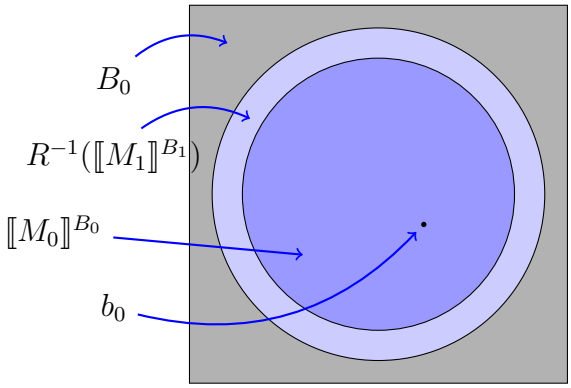


Figure 4.4: Heterogeneous abstraction using behavior relations.

Heterogeneous abstraction using behavior relations is illustrated in a case study in Chapter 6.

When using several abstractions of a model together towards system analysis, we can make use of a collection of models to abstract an underlying model. This is typically useful when there are different behaviors in different operating regimes that are best modeled by different models, where neither one fully represents the whole set of behaviors of the system, but their union does. This notion is made formal by the following definition.

Definition 4.3 (Model Coverage)**.** For a system model M_0 with a behavioral domain B_0, given a set of models M_i with corresponding behavior domains B_i and behavior relations $R_i \subseteq B_0 \times B_i$, models M_i, $i = 1, \ldots, n$ *cover* M_0 if there exists a partition $\{B_0^1, B_0^2, \ldots, B_0^n\}$ of $[\![M_0]\!]^{B_0}$ s.t. $\forall i = 1, 2, \ldots, n$

$$B_0^i \subseteq R_i^{-1}([\![M_i]\!]^{B_i}).$$

Fig. 4.5 provides a pictorial intuition behind the coverage definition from Def. 4.3. It ensures that every behavior of the underlying system M_0 to be accounted for by at least one model. Heterogeneous model coverage is illustrated in a case study in Sec. 6.3.

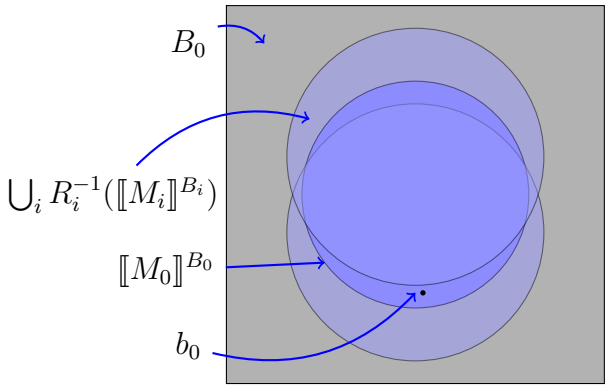


Figure 4.5: Heterogeneous coverage using behavior relations

A special case of coverage using multiple models occurs when the behavior of a system in different modes of operation can be best modeled using different models, and the system can switch between these modes. For safety specifications that require that some correctness condition always hold over all time (i.e., also over all modes no matter what order the system switches into them), the different models can be analyzed in isolation, provided we account for all possible manners of entering any given mode into the model for that particular mode. Heterogeneous model coverage for mode switching is illustrated in a case study in Sec. 6.3.

To address interdependencies between models across different formalisms, we use explicitly identified model parameters which will be used to define the interdependencies. We use the parametric semantic interpretations of models as defined in Sec. 3.4. The notions of model abstraction and coverage are extended to include parameter constraints as follows.

Definition 4.4 (Parametric Abstraction)**.** Given a parameterized model M_0 with a behavioral domain B_0, a parameterized model M_i with corresponding behavior formalism B_i and a behavior relation $R_i \subseteq B_0 \times B_i$, M_i is said to be a *parametric abstraction* of M_0 under an

auxiliary constraint C_{aux} if for any constraint C_0^M on P_0 such that C_{aux} is non-conflicting for C_0^M, for the effective external constraint $E_i^M := (C_{\mathsf{aux}} \wedge C_0^M) \downarrow_{P_i^M}$, we have

$$[\![C_0^M, M_0]\!]^{B_0} \subseteq R_i^{-1}([\![E_i^M, M_i]\!]^{B_i}). \tag{4.3}$$

In words, the above definition asserts that when the behaviors of the system model M_0 along with the system-level constraint C_0^M and a model M_i with the effective external constraint E_i^M due to C_i^M given the interdependencies C_{aux} are compared via the behavior relation R_i, the set of behaviors of M_i overapproximate that of M_0. Parametric abstraction is illustrated in Sec. 6.6.

Along similar lines as the definition of parametric abstraction, we define parametric coverage as follows.

Definition 4.5 (Parametric Coverage)**.** For a parameterized system model M_0 with a corresponding behavior formalism B_0, a given set of parameterized models M_i with corresponding behavior formalisms B_i and behavior relations $R_i \subseteq B_0 \times B_i$, $i = 1, \ldots, n$ form a *parametric cover* for M_0 if for any constraint C_0^M on P_0 such that C_{aux} is non-conflicting for C_0^M, there exists a partition $\{B_0^1, B_0^2, \ldots, B_0^n\}$ of $[\![C_0^M, M_0]\!]^{B_0}$ s.t. $\forall i = 1, 2, \ldots, n,$, for the effective external constraints $E_i^M := (C_{\mathsf{aux}} \wedge C_0^M) \downarrow_{P_i^M}$, we have

$$B_0^i \subseteq R_i^{-1}([\![E_i^M, M_i]\!]^{B_i}). \tag{4.4}$$

The intuition behind the definition of parametric coverage is similar to that of parametric abstraction, except that the models M_i with their effective external constraints E_i^M

only need to overapproximate their subset of the partition of the behaviors of the model M_0 with constraint C_0^M via their behavior relations R_i. Then because of the partition, the union of the model behaviors via R_i abstracts the behaviors of M_0.

4.2 Compositional Heterogeneous Abstraction

Models of complex systems are often composed of interacting component models. In such cases, heterogeneous abstraction can be independently established for the respective component models and these results can be used to infer heterogeneous abstraction for their composition [82]. We develop conditions under which heterogeneous abstraction between component models implies heterogeneous abstraction between the composite system models.

4.2.1 Heterogeneous Abstraction In Common Behavior Domains

Fig. 4.6 illustrates the compositional heterogeneous abstraction problem. We consider two levels of abstraction, represented by the subscripts $i = 0, 1$. For each level of abstraction, we assume there is a modeling formalism $\mathcal{M}_i$ and a behavior class $\mathcal{B}_i$. Component models $P_i, Q_i \in \mathcal{M}_i$ have their semantics defined in common behavior domains $B_i \in \mathcal{B}_i$. Behavior abstraction function $\mathcal{A} : B_0 \to B_1$ is used as a mapping between the behavior domains B_0 and B_1.

We use the definition of semantic composition from Def. 3.3 of two component models. The following proposition gives conditions for compositional heterogeneous abstraction.

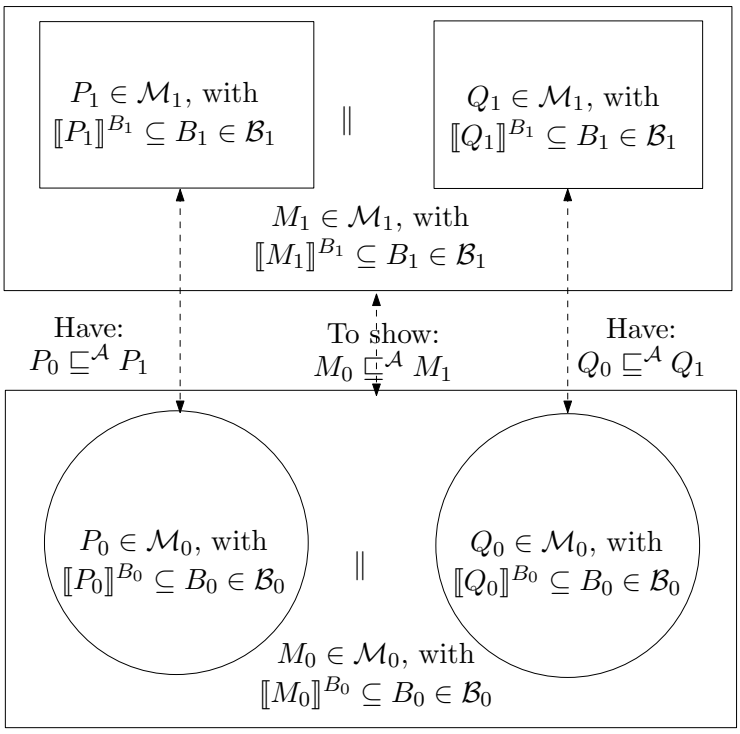


Figure 4.6: Compositional heterogeneous abstraction in common behavior domains.

Proposition 4.1. For each abstraction level $i = 0, 1$, given component models P_i, Q_i with the semantics of each model interpreted over a behavior domain B_i , and a behavior abstraction function $\mathcal{A} : B_0 \to B_1$, if $P_0 \sqsubseteq^{\mathcal{A}} P_1$ and $Q_0 \sqsubseteq^{\mathcal{A}} Q_1$, then

$$P_0||Q_0 \sqsubseteq^{\mathcal{A}} P_1||Q_1.$$

Proof. From $P_0 \sqsubseteq^{\mathcal{A}} P_1$ and $Q_0 \sqsubseteq^{\mathcal{A}} Q_1$, we have $[\![P_0]\!]^{B_0} \subseteq \mathcal{A}^{-1}([\![P_1]\!]^{B_1})$ and $[\![Q_0]\!]^{B_0} \subseteq$

$\mathcal{A}^{-1}([\![Q_1]\!]^{B_1})$. Therefore,

$$\begin{aligned}
[\![P_0||Q_0]\!]^{B_0} &= [\![P_0]\!]^{B_0} \cap [\![Q_0]\!]^{B_0} \\
&\subseteq \mathcal{A}^{-1}([\![P_1]\!]^{B_1}) \cap \mathcal{A}^{-1}([\![Q_1]\!]^{B_1}) \\
&= \mathcal{A}^{-1}([\![P_1]\!]^{B_1} \cap [\![Q_1]\!]^{B_1}) \\
&= \mathcal{A}^{-1}([\![P_1||Q_1]\!]^{B_1}).
\end{aligned}$$

□

This proposition states that with global semantics, composition of abstractions is the abstraction of the composition.

Remark 4.6 (Insufficiency of Behavior Relations)**.** We note that arbitrary behavior relations that are not functions are not sufficient in even this simple case of compositional heterogeneous abstraction. If a behavior relation $\mathcal{A}$ is not a function, it is possible to have a behavior $b_0 \in [\![P_0]\!]^{B_0} \cap [\![Q_0]\!]^{B_0}$ with $(b_0, p_1) \in \mathcal{A}$, $(b_0, q_1) \in \mathcal{A}$, s.t. $p_1 \in [\![P_1]\!]^{B_1} \backslash [\![Q_1]\!]^{B_1}$ and $q_1 \in [\![Q_1]\!]^{B_1} \backslash [\![P_1]\!]^{B_1}$ but $\nexists\ b_1 \in [\![P_1]\!]^{B_1} \cap [\![Q_1]\!]^{B_1}$ with $(b_0, b_1) \in \mathcal{A}$, as shown in the Venn diagram in Fig. 4.7.

For this b_0, we have $b_0 \in \mathcal{A}^{-1}([\![P_1]\!]^{B_1}) \cap \mathcal{A}^{-1}([\![Q_1]\!]^{B_1})$ but $b_0 \notin \mathcal{A}^{-1}([\![P_1]\!]^{B_1} \cap [\![Q_1]\!]^{B_1})$ and therefore the above proof does not hold. The arbitrary mappings that are the source of these counterexamples – those that allow one concrete behavior to be associated with more than one abstract behaviors – are perhaps not necessary in practice. The restriction from behavior relations to functions disallows the possibility of having several abstract behaviors correspond to a single concrete behavior, while still allowing several concrete

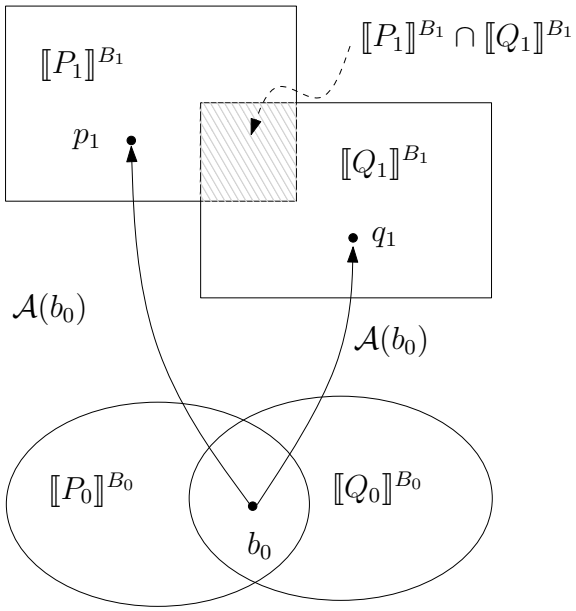


Figure 4.7: Insufficiency of arbitrary behavior relations for compositionality.

behaviors to be mapped to a single abstract behavior. □

Next we consider the general case where the local semantics of the two components are defined in terms of distinct behavior domains. Fig. 4.8 illustrates the general case of the compositional heterogeneous abstraction problem. Component models $P_i, Q_i \in \mathcal{M}_i$ have their semantics defined in terms of *local* behavior domains $B_i^P, B_i^Q \in \mathcal{B}_i$. These local domains include only the variables relevant to the given component. Because the local semantic domains B_i^P and B_i^Q for components P and Q are different from each other, their own behavior abstraction functions $\mathcal{A}^P$ and $\mathcal{A}^Q$ that are used as mappings between the respective local behavior domains. To compose the two models to form the system models $M_i \in \mathcal{M}_i$, the local semantics are lifted to *global* behavior domains $B_i \in \mathcal{B}_i$ to include variables from both components.

We begin by developing a mechanism to associate between different abstraction func-

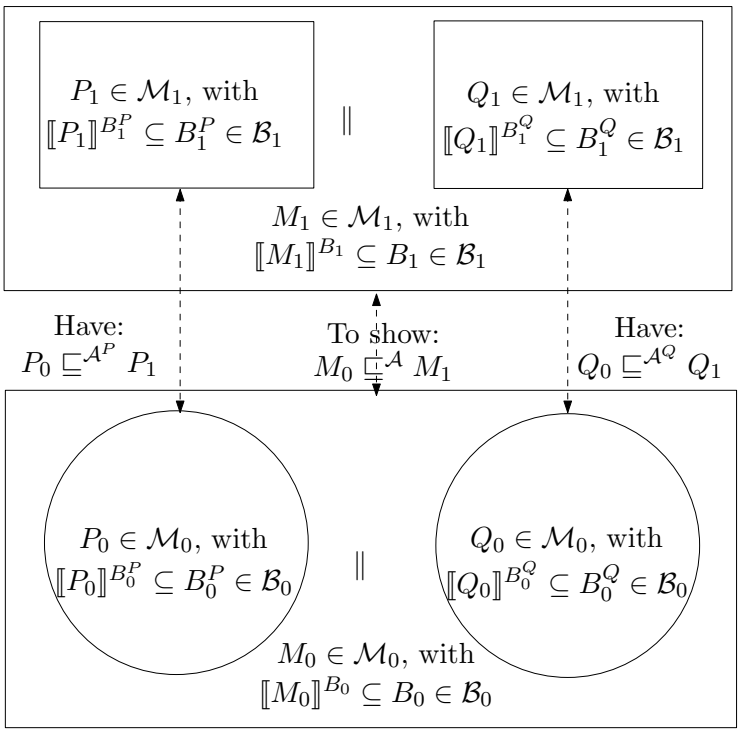


Figure 4.8: Compositional heterogeneous abstraction analysis in local behavior domains.

tions of component models.

4.2.2 Localization/Globalization of Abstraction Functions

We have defined behavior set localization and globalization as fundamental semantic operations in Chapter 3. Note that in case of compositional heterogeneous analysis as depicted in Fig. 4.8, there are four different behavior localizations (or globalizations) – one for each component and one at each level of abstraction. We index these with subscripts $i = 0, 1$ for the two levels of abstraction and superscripts $j = 1, 2$ or $j = P, Q$ to distinguish between these wherever necessary. Here we create the notions of localization/globalization of behavior abstraction functions given the localization/globalization mappings at *both* the levels of abstraction that the behavior abstraction function maps.

Given behavior globalizations at the abstract and concrete levels of abstraction, we next define the globalization of a behavior abstraction function between the abstract and concrete local domains.

Definition 4.7 (Abstraction Function Globalization)**.** Given two behavior classes $\mathcal{B}_0$ and $\mathcal{B}_1$, behavior domains from each behavior class: $B_0, B_0' \in \mathcal{B}_0$ and $B_1, B_1' \in \mathcal{B}_1$, localizations $\downarrow_i$ of B_i to B_i' for $i = 1, 2$, and a behavior abstraction function $\mathcal{A}'$ of B_0' to B_1', a behavior abstraction function $\mathcal{A}$ of B_0 to B_1 is said to be a *globalization* of $\mathcal{A}'$ if

$$\forall b_0 \in B_0 : \mathcal{A}'(b_0{\downarrow_0}) = \mathcal{A}(b_0){\downarrow_1}. \tag{4.5}$$

In words, the definition of abstraction globalization states that given any global concrete behavior b_0, the abstraction of its localization $b_0{\downarrow_0}$ at the concrete level 0 through the local abstraction function $\mathcal{A}'$ should be the same as the localization at the abstract level 1 of its corresponding abstract behavior $\mathcal{A}(b_0)$. This concept is illustrated by the diagram in Fig. 4.9: $\mathcal{A}$ is a globalization of $\mathcal{A}'$ if the diagram commutes.

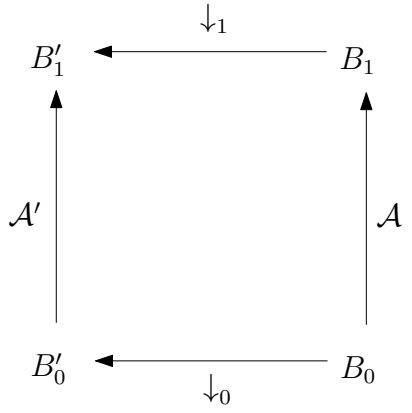


Figure 4.9: Commutative diagram for abstraction function localization/globalization.

We write $\mathcal{A} = \mathcal{A}'{\Uparrow}$ if $\mathcal{A}$ is a globalization of $\mathcal{A}'$. We call $\mathcal{A}'$ a *localization* of $\mathcal{A}$, written

$\mathcal{A}' = \mathcal{A}\Downarrow$, iff $\mathcal{A} = \mathcal{A}'\Uparrow$.

Note that in case of compositional heterogeneous analysis as depicted in Fig. 4.8, there are two different abstraction localizations/globalizations – one for each component.

We note the following existence and uniqueness properties of localization/globalization of behavior abstraction functions.

- **Existence of globalization.** For a given local abstraction $\mathcal{A}'$, it is always possible to construct a globalization $\mathcal{A}'\Uparrow$ s.t. the diagram commutes. This is due to the fact that both localizations $\downarrow_i$, $i = 0, 1$ are onto functions. Therefore, for any local behaviors b'_0 and $b'_1 = \mathcal{A}(b'_0)$, $b'_0\uparrow_0$ and $b'_1\uparrow_1$ are non-empty. One can then associate every behavior $b_0 \in b'_0\uparrow_0$ with some behavior $b_1 \in b'_1\uparrow_1$, which results in a valid globalization of $\mathcal{A}'$.
- **Non-uniqueness of globalization.** For a given local abstraction $\mathcal{A}'$, its globalization $\mathcal{A}'\Uparrow$ is not unique. For a b'_0 with $\mathcal{A}'(b'_0) = b'_1$ and $b'_1\uparrow_1 = \{b^0_1, b^1_1\}$, consider a global behavior $b_0 \in b'_0\uparrow_0$. Then $\mathcal{A}^0$ with $\mathcal{A}^0(b_0) = b^0_1$ and $\mathcal{A}^1$ with $\mathcal{A}^1(b_0) = b^1_1$ can both be globalizations of $\mathcal{A}'$. Since localization causes loss of information, its set-valued inverse provides some freedom for creating mappings at the global level; appropriate ones need to be chosen.
- **Non-existence of localization.** For a given global abstraction $\mathcal{A}$, its localization $\mathcal{A}\Downarrow$ may not exist, i.e., the diagram may not commute for any $\mathcal{A}'$. Consider b^0_0, b^1_0 with $\mathcal{A}(b^0_0) = b^0_1$ and $\mathcal{A}(b^1_0) = b^1_1$, and $b^0_0\downarrow_0 = b^1_0\downarrow_0$, but $b^0_1\downarrow_0 \neq b^1_1\downarrow_0$. For such a case, there can be no $\mathcal{A}'$ s.t. $\mathcal{A}'\Uparrow = \mathcal{A}$.
- **Uniqueness of localization.** For a given global abstraction $\mathcal{A}$, if $\mathcal{A}\Downarrow$ exists, it is unique. This is simply due to the diagram commuting. $\forall$ b_0, behaviors $b_0\downarrow_0$,

$\mathcal{A}(b_0) =: b_1$, and $b_1{\downarrow_1}$ are unique. Therefore, for every given mapping $\mathcal{A}(b_0) = b_1$, there is a unique mapping $\mathcal{A}'(b_0{\downarrow_0}) = b_1{\downarrow_1}$.

- **Globalization and localization are not necessarily inverse operations.** From the uniqueness of localization and non-uniqueness of globalization, it is straightforward to show that

$$(\mathcal{A}'{\Uparrow}){\Downarrow} = \mathcal{A}'; \tag{4.6}$$

but $(\mathcal{A}{\Downarrow}){\Uparrow}$ may not be equal to $\mathcal{A}$.

Given the theoretical machinery developed so far, we now find conditions under which compositional heterogeneous abstraction w.r.t. Fig. 4.8 can be used.

The following lemma states that heterogeneous abstraction between model globalizations via a global abstraction function is equivalent to heterogeneous abstraction between original models via the localization of the global abstraction function.

Lemma 4.1. For abstraction levels $i = 0, 1$, given component models P_i with local behavior domains B'_i, behavior localization functions ${\downarrow_i} : B_i \to B'_i$, let their corresponding globalized models be P_i^G with global behavior domains B_i. If $\mathcal{A} : B_0 \to B_1$ is a global behavior abstraction function and $\mathcal{A}' : B'_0 \to B'_1$ is a localization of $\mathcal{A}$, then

$$P_0^G \sqsubseteq^{\mathcal{A}} P_1^G \Leftrightarrow P_0 \sqsubseteq^{\mathcal{A}'} P_1.$$

Proof. From the definition of model globalization, we have

$$b_i \in [\![P_i^G]\!]^{B_i} \Leftrightarrow b_i{\downarrow_i} \in [\![P_i]\!]^{B'_i} \tag{4.7}$$

and

$$b'_i \in [\![P_i]\!]^{B'_i} \Leftrightarrow b'_i\uparrow_i \subseteq [\![P_i^G]\!]^{B_i}. \tag{4.8}$$

Case I: $P_0^G \sqsubseteq^{\mathcal{A}} P_1^G \Rightarrow P_0 \sqsubseteq^{\mathcal{A}'} P_1$

For any given $b_0 \in [\![P_0^G]\!]^{B_0}$, let $b_1 := \mathcal{A}(b_0)$. From $P_0^G \sqsubseteq^{\mathcal{A}} P_1^G$, we have $b_1 \in [\![P_1^G]\!]^{B_1}$. From (4.5), $\mathcal{A}'(b'_0 := b_0\downarrow_0) = b_1\downarrow_1$. Hence, from (4.7), we have that $\forall b'_0 \in [\![P_0]\!]^{B'_0}, \mathcal{A}'(b'_0) \in [\![P_1]\!]^{B'_1}$, which implies $[\![P_0]\!]^{B'_0} \subseteq \mathcal{A}'^{-1}([\![P_1]\!]^{B'_1})$, i.e., $P_0 \sqsubseteq^{\mathcal{A}'} P_1$.

Case II: $P_0^G \sqsubseteq^{\mathcal{A}} P_1^G \Leftarrow P_0 \sqsubseteq^{\mathcal{A}'} P_1$

From $P_0 \sqsubseteq^{\mathcal{A}'} P_1$, we have $b'_0 \in [\![P_0]\!]^{B'_0} \Rightarrow \mathcal{A}'(b'_0) =: b'_1 \in [\![P_1]\!]^{B'_1}$. From Def. 4.7 and (4.8), for any $b'_0 \in [\![P_0]\!]^{B'_0}$, $b_0 \in b'_0\uparrow_0 \subseteq [\![P_0^G]\!]^{B_0} \Rightarrow \mathcal{A}(b_0) =: b_1 \in b'_1\uparrow_1 \subseteq [\![P_1^G]\!]^{B_1}$. Therefore, $[\![P_0^G]\!]^{B_0} \subseteq \mathcal{A}^{-1}([\![P_1^G]\!]^{B_1})$, i.e., $P_0^G \sqsubseteq^{\mathcal{A}} P_1^G$. □

In terms of Fig. 4.8, the implication of Lemma 4.1 is the following. When the abstract and concrete models of a component are considered in isolation, it does not matter whether one does the heterogeneous abstraction analysis in the global domains or the local domains. We now use the result from Lemma 4.1 in a compositional setting when the component models are composed to form a system model.

4.2.3 Heterogeneous Abstraction In Local Behavior Domains

Consider the general scenario w.r.t. Fig. 4.8 in which the component models P_i and Q_i have different local behavior domains B_i^P and B_i^Q in behavior class $\mathcal{B}_i$, for levels of abstraction $i = 0, 1$. In this case, we need to lift the local semantics of the components to common global behavior domains before we can compose them. This operation is defined as model

globalization in Def. 3.6 as a fundamental operation on behavior sets.

For the following discussion, we let models M_i, with the global behavior domains B_i, be the globalized compositions $P_i||^G Q_i$ of component models P_i and Q_i with their local behavior domains B_i^P and B_i^Q, for levels of abstraction $i = 0, 1$ as depicted in Fig. 4.8. We consider two scenarios in which the source of the abstraction is at the system and component levels respectively.

Centralized development

First, we consider the case where an abstraction function $\mathcal{A} : B_0 \to B_1$ between the global behavior domains B_0 and B_1 is given. For this case, the following proposition shows that the problem of establishing $M_0 \sqsubseteq^{\mathcal{A}} M_1$ can be reduced to solving two smaller problems $P_0 \sqsubseteq^{\mathcal{A}\Downarrow^P} P_1$ and $Q_0 \sqsubseteq^{\mathcal{A}\Downarrow^Q} Q_1$.

Proposition 4.2. For abstraction levels $i = 0, 1$, given component models P_i and Q_i with corresponding local behavior domains B_i^P and B_i^Q, let their globalized semantic compositions be $P_i||^G Q_i$ in global behavior domains B_i with behavior localizations $\downarrow_i^j : B_i \to B_i^j$, where $j = P, Q$, and a global behavior abstraction function $\mathcal{A} : B_0 \to B_1$. If localizations $\mathcal{A}\Downarrow^P$ and $\mathcal{A}\Downarrow^Q$ of $\mathcal{A}$ exist and $P_0 \sqsubseteq^{\mathcal{A}\Downarrow^P} P_1$ and $Q_0 \sqsubseteq^{\mathcal{A}\Downarrow^Q} Q_1$, then $M_0 \sqsubseteq^{\mathcal{A}} M_1$.

Proof. From $P_0 \sqsubseteq^{\mathcal{A}^P} P_1$ and $Q_0 \sqsubseteq^{\mathcal{A}^Q} Q_1$, we know from Lemma 4.1 that $P_0^G \sqsubseteq^{\mathcal{A}} P_1^G$ and

$Q_0^G \sqsubseteq^{\mathcal{A}} Q_1^G$, i.e., that $[\![P_0^G]\!]^{B_0} \subseteq \mathcal{A}^{-1}([\![P_1^G]\!]^{B_1})$ and $[\![Q_0^G]\!]^{B_0} \subseteq \mathcal{A}^{-1}([\![Q_1^G]\!]^{B_1})$. We have,

$$\begin{aligned}
[\![P_0||^G Q_0]\!]^{B_0} &= [\![P_0^G]\!]^{B_0} \cap [\![Q_0^G]\!]^{B_0} \\
&\subseteq \mathcal{A}^{-1}([\![P_1^G]\!]^{B_1}) \cap \mathcal{A}^{-1}([\![Q_1^G]\!]^{B_1}) \\
&= \mathcal{A}^{-1}([\![P_1^G]\!]^{B_1} \cap [\![Q_1^G]\!]^{B_1}) \\
&= \mathcal{A}^{-1}([\![P_1||^G Q_1]\!]^{B_1}).
\end{aligned}$$

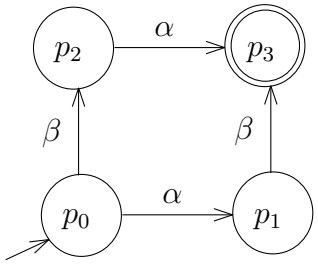
□

Prop. 4.2 states that we can establish $M_0 \sqsubseteq^{\mathcal{A}} M_1$ in the global behavior domains by establishing $P_0 \sqsubseteq^{\mathcal{A}\Downarrow^P} P_1$ and $Q_0 \sqsubseteq^{\mathcal{A}\Downarrow^Q} Q_1$ in the local behavior domains of the two components.

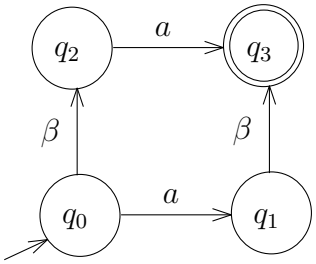
Example 14. Consider component models

$$P_0 \equiv \begin{bmatrix} \dot{x} \\ \\ \dot{y} \end{bmatrix} \in \begin{bmatrix} [2,4] \\ \\ [1,2] \end{bmatrix}, \begin{bmatrix} x \\ \\ y \end{bmatrix} (0) \in \begin{bmatrix} [0,l_x) \\ \\ [0,l_y) \end{bmatrix} \text{ and } Q_0 \equiv \begin{bmatrix} \dot{x} \\ \\ \dot{z} \end{bmatrix} \in \begin{bmatrix} [3,5] \\ \\ [1,2] \end{bmatrix}, \begin{bmatrix} x \\ \\ z \end{bmatrix} (0) \in \begin{bmatrix} [0,l_x) \\ \\ [0,l_z) \end{bmatrix}.$$

Let P_1 and Q_1 be as follows.

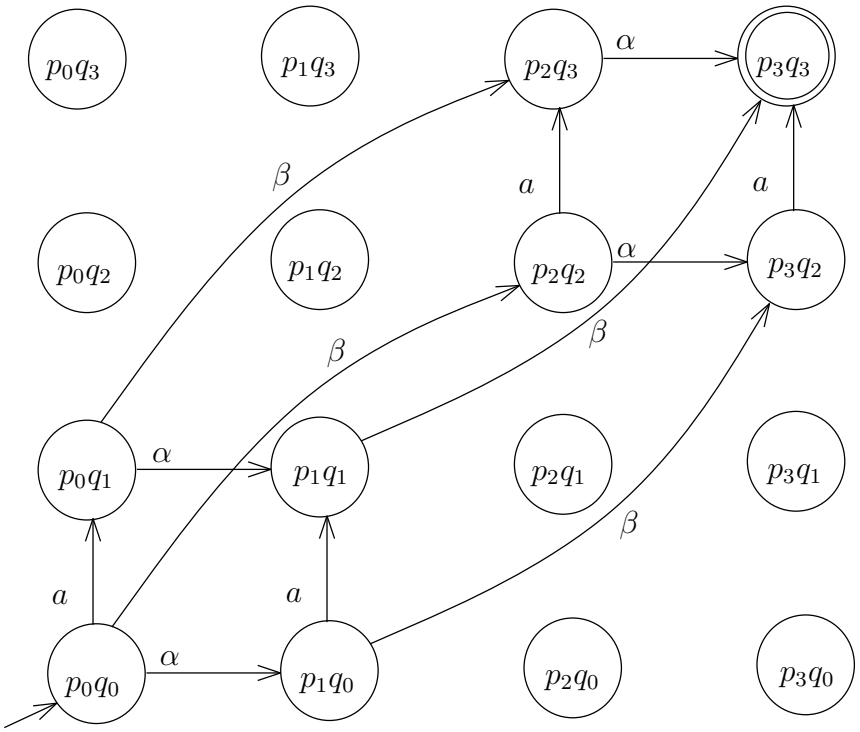


P_1



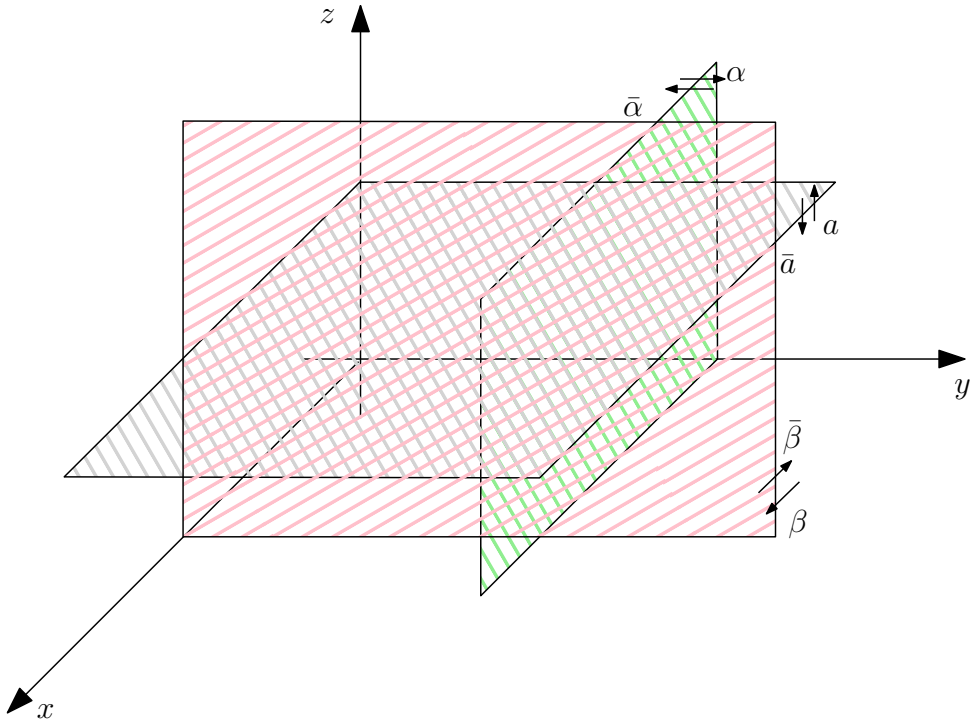
Q_1

The compositions are $M_0 := P_0||^G Q_0$ given by $\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \in \begin{bmatrix} [3,4] \\ [1,2] \\ [1,2] \end{bmatrix}, \begin{bmatrix} x \\ y \\ z \end{bmatrix} (0) \in \begin{bmatrix} [0,l_x) \\ [0,l_y) \\ [0,l_z) \end{bmatrix}$ and



$M_1 := P_1||^G Q_1$.

Global behavior domain $B_0 := (\mathbb{R}^3)^{\mathbb{R}_+}$ for M_0 is the set of 3-d trajectories with variable names x, y, z; while global behavior domain $B_1 := \Sigma^*$ for M_1 is the set of all finite traces over the alphabet $\Sigma = \{\alpha, \bar{\alpha}, \beta, \bar{\beta}, a, \bar{a}\}$. Let the behavior abstraction function $\mathcal{A} : B_0 \to B_1$ be defined by partitioning the continuous state space as follows.



Given $b_0 = [x(t)\ y(t)\ z(t)]^T =: \bar{x}(t)$, $t \in \mathbb{R}_+$ and $b_1 = \sigma_0\sigma_1\cdots$, $\mathcal{A}^j(b_0) = b_1$ if $\exists$ times $t_i \in \mathbb{R}_+$ s.t.

$$\forall t' \in [0, t_0), \qquad \bar{x}(t) \in \textsc{from}(\sigma_0),$$

$$\forall t' \in [t_{i-1}, t_i), \quad \bar{x}(t) \in \textsc{to}(\sigma_{i-1}) \cap \textsc{from}(\sigma_i),$$

$$\forall t' \geq t_N \qquad \bar{x}(t) \in \textsc{to}(\sigma_N)$$

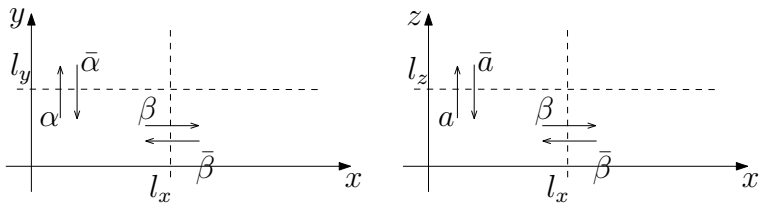
where $i = 1, \ldots, N$ for some $N \in \mathbb{N}$ and $\textsc{from}(\cdot)$ and $\textsc{to}(\cdot)$ are given in the following table.

σ	FROM(σ)	TO(σ)
a	$z \leq l_z,\ x, y \in \mathbb{R}$	$z \geq l_z,\ x, y \in \mathbb{R}$
$\bar{a}$	$z \geq l_z,\ x, y \in \mathbb{R}$	$z \leq l_z,\ x, y \in \mathbb{R}$
α	$y \leq l_y,\ x, z \in \mathbb{R}$	$y \geq l_y,\ x, z \in \mathbb{R}$
$\bar{\alpha}$	$y \geq l_y,\ x, z \in \mathbb{R}$	$y \leq l_y,\ x, z \in \mathbb{R}$
β	$x \leq l_x,\ y, z \in \mathbb{R}$	$x \geq l_x,\ y, z \in \mathbb{R}$
$\bar{\beta}$	$x \geq l_x,\ y, z \in \mathbb{R}$	$x \leq l_x,\ y, z \in \mathbb{R}$

Otherwise, $\mathcal{A}(b_0) = \varepsilon$.

The problem of establishing $M_0 \sqsubseteq^{\mathcal{A}} M_1$ for above $\mathcal{A}$ can be reduced to two smaller problems $P_0 \sqsubseteq^{\mathcal{A}\Downarrow^P} P_1$ and $Q_0 \sqsubseteq^{\mathcal{A}\Downarrow^Q} Q_1$ as follows.

Local behavior domains for the two models of component P are $B_0^P = (\mathbb{R}^2)^{\mathbb{R}_+}$ with variable names for the dimensions x and y; and $B_1^P = \Sigma^{P*}$ with $\Sigma^P = \{\alpha, \bar{\alpha}, \beta, \bar{\beta}\}$. Similarly, in case of the two models of component Q, $B_0^Q = (\mathbb{R}^2)^{\mathbb{R}_+}$ with variable names for the dimensions x and z; and $B_1^Q = \Sigma^{Q*}$ with $\Sigma^Q = \{a, \bar{a}, \beta, \bar{\beta}\}$. Let behavior localization functions for the two components at the two levels of abstractions be variable elimination and natural projection as in Ex. 8 and 9. We get the behavior abstraction function localizations $\mathcal{A}^P : B_0^P \to B_1^P$ and $\mathcal{A}^Q : B_0^Q \to B_1^Q$, where $\mathcal{A}^P = \mathcal{A}\Downarrow^P$ and $\mathcal{A}^Q = \mathcal{A}\Downarrow^Q$ are as follows.



Let $\bar{x}^P = [xy]^T$ and $\bar{x}^Q = [xz]^T$. Given $b_0^P = \bar{x}^P(t)$, $b_0^Q = \bar{x}^Q(t)$ for $t \in \mathbb{R}_+$ and $b_1^j = \sigma_0^j \sigma_1^j \cdots$, $\mathcal{A}^j$, $j = P, Q$, are defined as $\mathcal{A}^j(b_0^j) = b_1^j$ if $\exists$ times $t_i^j \in \mathbb{R}_+$ s.t.

$$\forall t' \in [0, t_0^j), \qquad \bar{x}^j(t') \in \textsc{from}^j(\sigma_0^j),$$

$$\forall t' \in [t_{i-1}^j, t_i^j), \quad \bar{x}^j(t') \in \textsc{to}^j(\sigma_{i-1}^j) \cap \textsc{from}^j(\sigma_i^j),$$

$$\forall t' \geq t_N^j, \qquad \bar{x}^j(t') \in \textsc{to}^j(\sigma_N^j),$$

where $i = 1, \ldots, N$ for some $N \in \mathbb{N}$ and $\textsc{from}^j(\cdot)$ and $\textsc{to}^j(\cdot)$ are given in the following tables.

σ	$\textsc{from}^P(\sigma)$	$\textsc{to}^P(\sigma)$
α	$y \leq l_y, x \in \mathbb{R}$	$y \geq l_y, x \in \mathbb{R}$
$\bar{\alpha}$	$y \geq l_y, x \in \mathbb{R}$	$y \leq l_y, x \in \mathbb{R}$
β	$x \leq l_x, y \in \mathbb{R}$	$x \geq l_x, y \in \mathbb{R}$
$\bar{\beta}$	$x \geq l_x, y \in \mathbb{R}$	$x \leq l_x, y \in \mathbb{R}$

σ	$\textsc{from}^Q(\sigma)$	$\textsc{to}^Q(\sigma)$
a	$z \leq l_z, x \in \mathbb{R}$	$z \geq l_z, x \in \mathbb{R}$
$\bar{a}$	$z \geq l_z, x \in \mathbb{R}$	$z \leq l_z, x \in \mathbb{R}$
β	$x \leq l_x, z \in \mathbb{R}$	$x \geq l_x, z \in \mathbb{R}$
$\bar{\beta}$	$x \geq l_x, z \in \mathbb{R}$	$x \leq l_x, z \in \mathbb{R}$

Otherwise, $\mathcal{A}^j(b_0^j) = \varepsilon$.

From the initial conditions and the monotonicity of the dynamics of P_0 (resp. Q_0), we can see that every behavior of the concrete model crosses the $x == l_x$ and $y == l_y$ (resp. $z == l_z$) boundaries in either order and have corresponding behaviors $\alpha\beta$ (resp. $a\beta$) or $\beta\alpha$

(resp. βa) at the discrete level that they map to. Therefore $P_0 \sqsubseteq^{\mathcal{A}^P} P_1$ and $Q_0 \sqsubseteq^{\mathcal{A}^Q} Q_1$. Using Prop. 4.2, $M_0 \sqsubseteq^{\mathcal{A}} M_1$.

Here, analyzing P_is and Q_is is much easier than analyzing M_is directly. In general, the extent of savings achieved by doing the heterogeneous abstraction analysis compositionally depends on how much smaller the local behavior domains are compared to the global ones. □

Decentralized development

Now, we consider the case where the abstraction functions $\mathcal{A}^P : B_0^P \to B_1^P$ and $\mathcal{A}^Q : B_0^Q \to B_0^Q$ between the local behavior domains B_i^P and B_i^Q are given and heterogeneous abstractions of component models $P_0 \sqsubseteq^{\mathcal{A}^P} P_1$ and $Q_0 \sqsubseteq^{\mathcal{A}^Q} Q_1$ are established independently. This is the more common situation in practice, particularly for distributed development. In this case, the following proposition states that if the globalizations of abstraction functions $\mathcal{A}^P \Uparrow^P$ and $\mathcal{A}^Q \Uparrow^Q$ are defined consistently, the heterogeneous abstraction results for the components carry over to their compositions.

Proposition 4.3. For abstraction levels $i = 0, 1$, given component models P_i and Q_i with local behavior domains B_i^P and B_i^Q, let their compositions be $P_i||^G Q_i$ in global behavior domains B_i and local behavior abstraction functions be $\mathcal{A}^P : B_0^P \to B_1^P$ and $\mathcal{A}^Q : B_0^Q \to B_1^Q$ s.t. $P_0 \sqsubseteq^{\mathcal{A}^P} P_1$ and $Q_0 \sqsubseteq^{\mathcal{A}^Q} Q_1$. If $\mathcal{A}^P \Uparrow^P = \mathcal{A}^Q \Uparrow^Q =: \mathcal{A}$, i.e., then $P_0||^G Q_0 \sqsubseteq^{\mathcal{A}} P_1||^G Q_1$.

Proof. The result follows due to $(\mathcal{A}^P \Uparrow^P)\Downarrow^P = \mathcal{A}^P$ and $(\mathcal{A}^Q \Uparrow^Q)\Downarrow^Q = \mathcal{A}^Q$ from (4.6) and Prop. 4.2. □

Prop. 4.3 states that the heterogeneous abstraction results for component models P_i

and Q_i via possibly very different abstraction functions $\mathcal{A}^P$ and $\mathcal{A}^Q$ follow over to the system heterogeneous abstraction so long as $\mathcal{A}^P$ and $\mathcal{A}^Q$ are consistent, i.e., that it is possible to find globalizations $\mathcal{A}^P{\Uparrow}^P$ and $\mathcal{A}^Q{\Uparrow}^Q$ that are in agreement with each other. Note from the non-uniqueness of globalization of abstraction functions that there is some design freedom while constructing the semantic mappings at the global behavior domains for the two components such that they agree.

We note the following conditions for agreement of the globalizations of the local abstraction functions from the two components.

- **Disjoint behavior domains.** If the local behavior domains are disjoint (no common variables), the abstraction functions are disjoint. Therefore, when globalized, they are not mutually restrictive and it is always possible to construct globalizations that agree.
- **Agreement in intersection.** For non-disjoint local behavior domains, it is necessary for globalization agreement that the local abstraction functions agree on the "intersection" of the two behavior domains, say $B_i^{\cap}$, i.e., along the variables common to the two components. If localizations $\mathcal{A}^P{\Downarrow}^{\cap} : B_0^{\cap} \to B_1^{\cap}$ and $\mathcal{A}^Q{\Downarrow}^{\cap} : B_0^{\cap} \to B_1^{\cap}$ of $\mathcal{A}^P$ and $\mathcal{A}^Q$ agree, it is always possible to construct globalizations of $\mathcal{A}^P$ and $\mathcal{A}^Q$ that agree due to the fact that variables not common to the two components are not mutually constraining.

Decentralized compositional heterogeneous abstraction is illustrated using a case study in Sec. 6.5.

4.3 Summary

This chapter presents a mechanism to establish abstraction between models in different formalisms using behavioral semantics. When possible, the semantic interpretations of heterogeneous models can be defined in a common behavior domain, which enables the use of the notion of abstraction using behavior set inclusion. When model semantics are defined in different behavior domains, associations between these domains using behavior relations enable us to relate the corresponding behavior sets.

When models are composed of interacting component or subsystem models, heterogeneous abstraction can be established independently for each component in isolation in its local behavior domains. Behavior abstraction functions, special cases of behavior relations, are used as associations between heterogeneous behavior domains at different levels of abstraction; while localizations/globalizations are used as associations between local component behavior domains and global system behavior domains at a given level of abstraction. Sufficient conditions are developed under which heterogeneous abstraction between component models implies heterogeneous abstraction between their compositions.

Behavior relations are general mappings for associating behaviors from two different domains. They are useful when associating pairs of behaviors where there is no clear notion of one being more abstract than the other. Behavior abstraction functions, on the other hand, are mappings defined with the purpose of establishing abstraction in mind. Behavior abstraction functions prohibit one concrete behavior to be mapped to more than one abstract behaviors, which is in accordance with what is needed for abstraction. Behavior relations allow more than one behaviors from one domain to be mapped to more than one

behaviors from the other. Behavior relations can also leave certain behaviors unmapped if they are not relevant for the analyses at hand. Behavior abstraction functions need to associate mappings to every behavior since they are used for the goal of abstraction. The appropriate choice of mappings depends on the particular domains and the behavior association objective at hand.

Given the mechanism developed in this chapter to associate and compare sets of behaviors and semantic interpretations of heterogeneous models using behavior relations and behavior abstraction functions, in the next chapter we develop heterogeneous verification across different semantic domains.

Chapter 5

Heterogeneous Verification

The goal of heterogeneous verification in model-based design is to infer properties about an underlying system by using its models or abstractions in different modeling formalisms and analyzing them for correctness against specifications using relevant analysis procedures and tools. In this chapter, we introduce the concept of formal specifications over heterogeneous behavior domains and their use along with heterogeneous model abstraction and coverage concepts developed in the previous chapter towards hierarchical heterogeneous formal verification of systems.

5.1 Specification Implication Using Behavioral Semantics

We begin by establishing semantic relationships between specifications that are similar to the semantic tools developed for abstraction and coverage between models.

Definition 5.1 (Specification Implication)**.** When semantically interpreted over the same set of behaviors B, a (stronger) specification S_1 is said to *imply* a (weaker) specification S_0, written $S_1 \Rightarrow^B S_0$, if

$$[\![S_1]\!]^B \subseteq [\![S_0]\!]^B. \tag{5.1}$$

This definition simply asserts that any behavior that satisfies S_1 also satisfies S_0. In this definition, note that the two specifications can be from heterogeneous specification formalisms $\mathcal{S}_0$ and $\mathcal{S}_1$, so long as their semantics are defined in the same behavior domain.

Example 15. Consider two cars crossing through an intersection as shown in Fig. 5.1. Consider an English language specification S_0 stated as "`the two cars never collide in the intersection`," and a temporal logic specification S_1 written as $\Box\neg(x_{\texttt{red}} \in [0, f] \wedge y_{\texttt{green}} \in [0, h])$, which requires that the two cars cannot be in the intersection at the same time.

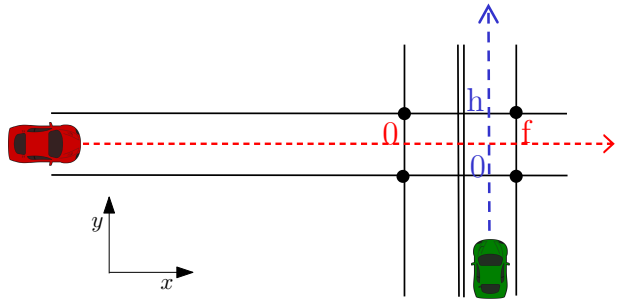


Figure 5.1: Intersection with two cars crossing.

Consider a behavior domain B, which is the set of all 4-d continuous trajectories in the variables $x_{\texttt{red}}, y_{\texttt{red}}, x_{\texttt{green}}, y_{\texttt{green}}$ for the x, y positions of the red and green car, such that $y_{\texttt{red}} \in [0, h]$ and $x_{\texttt{green}} \in [0, f]$ always hold, i.e., that the cars are always within their road

limits. The set $[\![S_0]\!]^B$ permits the cars to be in the intersection area so long as they do not collide (which depending on the intended interpretation of collide might mean 'be within ε' distance of each other for some ε). The set $[\![S_1]\!]^B$ simply does not allow the two cars to be in the intersection at the same time, which is more restrictive. Therefore we have $S_1 \Rightarrow^B S_0$. □

The following definition extends the notion of specification implication to heterogeneous behavior domains using behavior relations.

Definition 5.2 (Heterogeneous Implication)**.** Given behavior domains B_0, B_1 in behavior formalisms $\mathcal{B}_0$ and $\mathcal{B}_1$, and a behavior relation $R \subseteq B_0 \times B_1$, we say that specification S_1 *implies* specification S_0 *via* R, written $S_1 \Rightarrow^R S_0$, if

$$R^{-1}([\![S_1]\!]^{B_1}) \subseteq [\![S_0]\!]^{B_0}. \tag{5.2}$$

Fig. 5.2 shows a Venn diagram representation for the heterogeneous implication definition using behavioral semantics. It asserts that if a behavior $b_0 \in B_0$ is associated through R with a behavior in $b_1 \in B_1$ that satisfies S_1, then b_0 satisfies S_0.

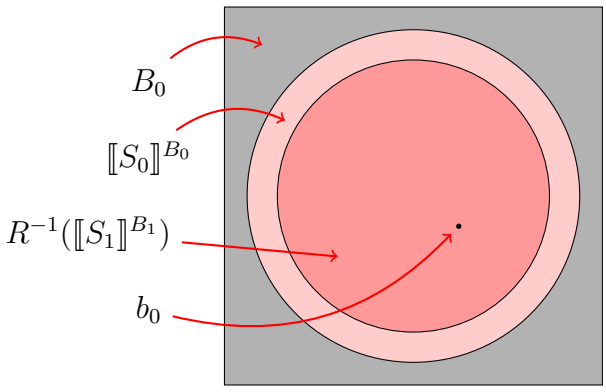


Figure 5.2: Heterogeneous implication using behavior relations.

When several specifications are used to define correctness requirements for different models, we define conjunction of specifications to be the intersection of the behavior sets allowed by each of the specifications. We need to ensure that the specifications checked against each model together imply the specification for the underlying system. The following definition makes this notion formal.

Definition 5.3 (Conjunctive Heterogeneous Implication)**.** Given system behavior domain B_0, behavior domains B_i and behavior relations $R_i \subseteq B_0 \times B_i$, $i = 1, \ldots, n$, specifications $S_i, i = 1, \ldots, n$, *conjunctively imply* the system specification S_0 if

$$\bigcap_i R_i^{-1}([\![S_i]\!]^{B_i}) \subseteq [\![S_0]\!]^{B_0}.$$

Fig. 5.3 shows a Venn diagram representation of conjunctive heterogeneous implication using behavior semantics. The definition allows the individual specifications S_i to not imply S_0, but their conjunction (intersection of the allowed behaviors) is required to be stronger than S_0.

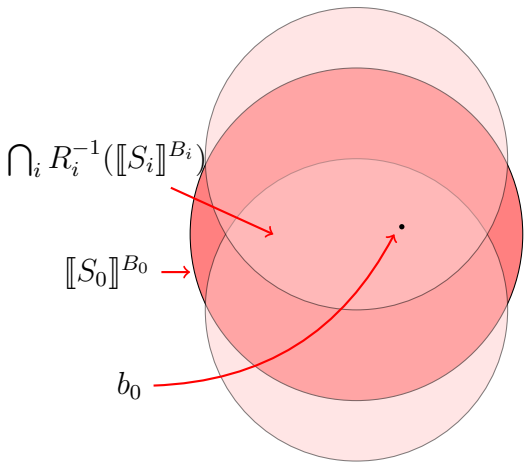


Figure 5.3: Conjunctive heterogeneous implication using behavior relations.

When there are interdependencies between specifications for different models, we explicitly model them using constraints over parameters and use parametric semantics for specifications as developed in Sec. 3.4. We extend the notion of specification implication and conjunctive specification implication to include parameter constraints as follows.

Definition 5.4 (Parametric Implication)**.** Given a behavior relation $R_i \subseteq B_0 \times B_i$ and an auxiliary constraint C_{aux}, a parameterized specification S_i is said to *parametrically imply* a parameterized specification S_0 if for any constraint C_0^S on P_0 such that C_{aux} is non-conflicting for C_0^S, for the effective external constraint $E_i^S := (C_{\mathsf{aux}} \wedge C_0^S) \downarrow_{P_i^S}$, we have

$$R_i^{-1}([\![E_i^S, S_i]\!]^{B_i}) \subseteq [\![C_0^S, S_0]\!]^{B_0}. \tag{5.3}$$

In words, the definition of parametric implication states that when the system-level specification S_0 with its system-level constraint C_0^S and specification S_i with its effective external constraint E_i^S due to S_0 given the interdependencies C_{aux} are compared, the behaviors allowed by S_i underapproximate those allowed by S_0. Note the similarity of this definition to the analogous Def. 4.4 for models.

The following definition develops the parametric extension of conjunctive specificaion implication.

Definition 5.5 (Conjunctive Parametric Implication)**.** For a parameterized system specification S_0 with a set of parameters P_0 and a corresponding behavioral formalism B_0, a given set of parameterized specifications S_i with corresponding behavior formalisms B_i and behavior relations $R_i \subseteq B_0 \times B_i$, S_i, $i = 1, \ldots, n$, *conjunctively paremetrically imply* S_0

if for any constraint C_0^S on P_0 such that C_{aux} is non-conflicting for C_0^S, for the effective external constraints $E_i^S := (C_{\mathsf{aux}} \wedge C_0^S) \downarrow_{P_i^S}$, we have

$$\bigcap_i R_i^{-1}([\![E_i^S, S_i]\!]^{B_i}) \subseteq [\![C_0^S, S_0]\!]^{B_0}. \tag{5.4}$$

The intuition behind conjunctive parametric implication is similar to that of the parametric implication except that the individual specifications S_i with their effective external constraints E_i^S are allowed to not have to individually imply S_0 for its constraint C_0^S, although the conjunction of the specifications has to. Note the similarity of this definition to the analogous Def. 4.5 for models.

Conjunctive parametric implication is illustrated in a case study in Sec. 6.6.

5.2 Verification Using a Single Heterogeneous Abstraction

We begin the development of heterogeneous verification with a simple case using a single abstract model, which serves as a basic building block for multi-model system-level verification. Given the definitions of heterogeneous abstraction and heterogeneous implication, we develop the following proposition for heterogeneous verification that uses both of these concepts.

Proposition 5.1 (Heterogeneous Verification)**.** Given two behavior domains B_0 and B_1 in behavior formalisms $\mathcal{B}_0$ and $\mathcal{B}_1$, models M_0 and M_1 in modeling formalisms $\mathcal{M}_0$ and

$\mathcal{M}_1$, specifications S_0 and S_1 in specification formalisms $\mathcal{S}_0$ and $\mathcal{S}_1$, and a behavior relation $R \subseteq B_0 \times B_1$, if $M_0 \sqsubseteq^R M_1$, $M_1 \models^{B_1} S_1$ and $S_1 \Rightarrow^R S_0$, then $M_0 \models^{B_0} S_0$.

Proof. From $M_0 \sqsubseteq^R M_1$, we have

$$\begin{aligned} [\![M_0]\!]^{B_1} &\subseteq R^{-1}([\![M_1]\!]^{B_1}) \\ (\text{From } M_1 \models^{B_1} S_1) &\subseteq R^{-1}([\![S_1]\!]^{B_1}) \\ (\text{From } S_1 \Rightarrow^R S_0) &\subseteq [\![S_0]\!]^{B_0}. \end{aligned}$$

Therefore, $M_0 \models^{B_0} S_0$. □

The following diagram shows the schematic of heterogeneous verification according to Prop. 5.1.

M_1	$\models^{B_1}$	S_1
$\sqsubseteq^R$		$\Downarrow^R$
M_0	$\models^{B_0}$	S_0

The abstraction and implication relations between the respective models and specifications via the behavior relation R give us the ability to use the analysis result $M_1 \models^{B_1} S_1$ at an abstract level in behavior domain B_1 to conclude $M_0 \models^{B_0} S_0$ in a detailed behavior domain B_0.

5.3 Verification Using Several Heterogeneous Abstractions

There are two natural ways of using *multiple* models and specifications together. In one, models individually are abstractions of the underlying system and the conjunction of their associated specifications needs to imply the system specification. Alternatively, each model may represent only a subset of the behaviors of the underlying system, and the collection of models provides an abstraction of the complete system. In this second case, the specification for each model needs to imply the specification of interest for the underlying system for the set of behaviors covered by the model. We develop these two notions in the context of heterogeneous verification.

5.3.1 Conjunctive Multi-Model Heterogeneous Verification

We first consider the case where each model in a collection of models is a heterogeneous abstraction of the underlying system and specifications for the set of models together form a conjunctive implication for the system specification. In this case, we have the following analysis result.

Proposition 5.2 (Heterogeneous Conjunctive Analysis)**.** For a system model M_0 with a behavioral domain B_0 and specification S_0, given models M_i with the corresponding behavior domains B_i, specifications S_i and behavior relations $R_i \subseteq B_0 \times B_i$, if $M_0 \sqsubseteq^{R_i} M_i$, specifications S_i conjunctively imply S_0, and $M_i \models^{B_i} S_i$ for each $i = 1, \ldots, n$, then $M_0 \models^{B_0} S_0$.

Proof. From $M_0 \sqsubseteq^{R_i} M_i$ for each i, we have

$$\begin{aligned} [\![M_0]\!]^{B_0} &\subseteq \bigcap_i R_i^{-1}([\![M_i]\!]^{B_i}) \\ (\text{since } M_i \models^{B_i} S_i) &\subseteq \bigcap_i R_i^{-1}([\![S_i]\!]^{B_i}) \\ (\text{Conj. Het. Implication}) &\subseteq [\![S_0]\!]^{B_0}. \end{aligned}$$

Therefore, $M_0 \models^{B_0} S_0$. □

Fig. 5.4 gives a pictorial intuition for the heterogeneous conjunctive analysis from Prop. 5.2 using a Venn diagram representation of various behavior sets in the domain B_0. Model M_0 individually abstracted by models M_i means that its behavior set lies in the intersection of those of M_i via behavior relations R_i. Each model satisfying their specifications means their behavior sets are inside respected behavior sets of the specifications S_i. Finally, conjunctive heterogeneous implication says that the intersection of the behavior sets of specifications S_i via R_i is contained inside that of S_0. Conjunctive heterogeneous verification is illustrated in a case study in Sec. 6.4.

5.3.2 Disjunctive Multi-Model Heterogeneous Verification

In the disjunctive case, no model is a proper abstraction of the underlying system, only all models together cover it. Hence, in order to make sure that a specification holds for the underlying system we need to verify that each of the disjunctive models satisfies that specification.

From the definition of model coverage, we have the following lemma.

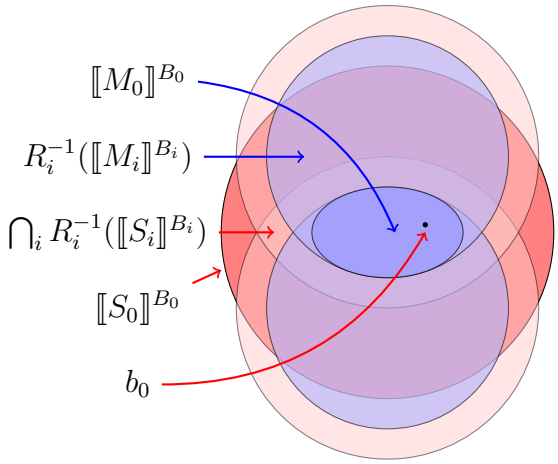


Figure 5.4: Conjunctive heterogeneous verification using behavior relations.

Lemma 5.1. If models M_i cover M_0 through R_i, $i = 1, \ldots, n$, we have

$$[\![M_0]\!]^{B_0} \subseteq \bigcup_{i=1}^{n} R_i^{-1}([\![M_i]\!]^{B_i}).$$

Proof. From the definition of partition, we have

$$\begin{aligned} [\![M_0]\!]^{B_0} &= \bigcup_{i=1}^{n} B_0^i \\ \text{(Def. 4.3)} \quad &\subseteq \bigcup_{i=1}^{n} R_i^{-1}([\![M_i]\!]^{B_i}). \end{aligned}$$

□

We use this lemma in heterogeneous disjunctive analysis as follows.

Proposition 5.3 (Heterogeneous Disjunctive Analysis)**.** For system model M_0 with a behavioral domain B_0 and specification S_0, given models M_i with the corresponding behavior domains B_i, specifications S_i and behavior relations $R_i \subseteq B_0 \times B_i$, if each specification

S_i heterogeneously implies S_0, models M_i cover M_0, and $M_i \models^{B_i} S_i$ for each $i = 1, \ldots, n$, then $M_0 \models^{B_0} S_0$.

Proof. From the definition of model coverage, we have

$$\begin{aligned} [\![M_0]\!]^{B_0} &\subseteq \bigcup_i R_i^{-1}([\![M_i]\!]^{B_i}) \\ (\text{since } M_i \models^{B_i} S_i) &\subseteq \bigcup_i R_i^{-1}([\![S_i]\!]^{B_i}) \\ (\text{Het. Implication}) &\subseteq [\![S_0]\!]^{B_0}. \end{aligned}$$

Therefore, $M_0 \models^{B_0} S_0$. □

Fig. 5.5 gives a pictorial intuition behind the disjunctive heterogeneous analysis construct from Prop. 5.3. The union of the behaviors for models M_i via behavior relations R_i cover the set of behaviors for M_0 in behavior domain B_0. Individual safety verification results mean that the sets of behaviors of models M_i are in those of S_i. Finally, each specification S_i being stronger than S_0 via R_i means that the union of the behavior sets for S_i via R_i is contained inside the set of behaviors allowed by S_0 in B_0. Disjunctive heterogeneous analysis is illustrated in a case study in Sec. 6.3.

5.4 Consistent Heterogeneous Verification with Interdependencies

To ensure consistent heterogeneous verification in presence of ineterdependencies, we extend the notions of model abstraction and coverage in Sec. 4.1 as well as specification

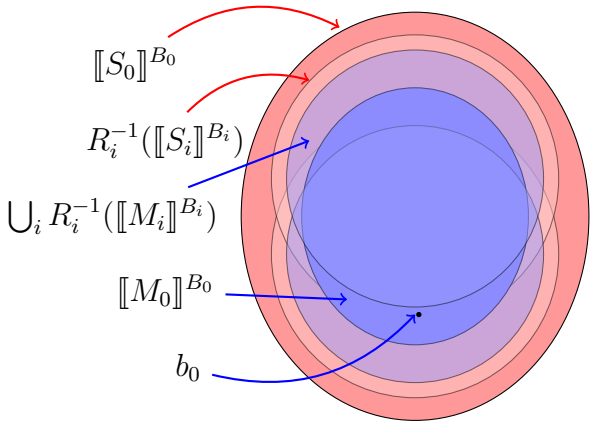


Figure 5.5: Disjunctive heterogeneous verification using behavior relations

implication in Sec. 5.1 to include constraints over parameters. We begin by developing a notion of *external-constraint consistency*, which ensures that the values of parameters used for each analysis task correctly approximates original system-level parameter valuations given the interdependencies.

Definition 5.6. The pair of constraints (C_i^M, C_i^S) for i^{th} analysis task $C_i^M, M_i \models^{B_i} C_i^S, S_i$ is said to be *external-constraint consistent* if

$$E_i^M := (C_0^M \wedge C_{\mathsf{aux}}) \downarrow_{P_i^M} \Rightarrow C_i^M \text{ and } C_i^S \Rightarrow (C_0^S \wedge C_{\mathsf{aux}}) \downarrow_{P_i^S} =: E_i^S. \tag{5.5}$$

The constraints E_i^M and E_i^S capture the effective external constraints on the local sets of parameters P_i^M and P_i^S; while the constraints C_i^M and C_i^S get used for establishing parametric entailment for model M_i and specification S_i. The external-constraint consistency ensures that the constraints used for individual analysis are over- and under-approximative in the correct sense. The monotonicity of parameterization of parameters then ensures that the corresponding model and specification behavior sets are also over-

and under-approximated in the correct manner.

Given these definitions, the following two propositions give sufficient conditions for parametric conjunctive and disjunctive analysis.

Proposition 5.4. Given parameterized system model M_0 and specification S_0 with corresponding behavior domain B_0 and the pair of constraints C_0^M, C_0^S over the system-level parameters P_0^M, P_0^S, a set of parameterized models M_i and specifications S_i with corresponding behavior formalisms B_i, behavior relations $R_i \subseteq B_0 \times B_i$ and pairs of constraints C_i^M, C_i^S over parameters P_i^M, P_i^S for $i = 1, \ldots, n$, if

i. constraints (C_i^M, C_i^S) are external-constraint consistent,

ii. each model M_i is a parametric abstraction of M_0,

iii. specifications S_i conjunctively parametrically imply S_0, and

iv. $C_i^M, M_i \models^{B_i} C_i^S, S_i$

then $C_0^M, M_0 \models^{B_0} C_0^S, S_0$.

Proof. From the definition of parametric abstraction, we have

$$\begin{aligned}
[\![C_0^M, M_0]\!]^{B_0} &= \bigcap_i R_i^{-1}([\![E_i^M, M_i]\!]^{B_i}) \\
\text{(Def. 5.6, monotonicity)} &\subseteq \bigcap_i R_i^{-1}([\![C_i^M, M_i]\!]^{B_i}) \\
(C_i^M, M_i \models^{B_i} C_i^S, S_i) &\subseteq \bigcap_i R_i^{-1}([\![C_i^S, S_i]\!]^{B_i}) \\
\text{(Def. 5.6, monotonicity)} &\subseteq \bigcap_i R_i^{-1}([\![E_i^S, S_i]\!]^{B_i}) \\
\text{(Def. 5.5)} &\subseteq [\![C_0^S, S_0]\!]^{B_0}
\end{aligned}$$

Therefore, $C_0^M, M_0 \models^{B_0} C_0^S, S_0$. □

This proposition presents the parametric counterpart of the conjunctive heterogeneous analysis construct from Prop. 5.2. It uses the parametric definitions of abstraction (Def. 4.4), entailment (Eq. (3.2)) and conjunctive implication (Def. 5.5). The external-constraint consistency and monotonicity provide the remaining pieces for the conjunctive parametric verification to work out. Conjunctive parametric verification is illustrated in a case study in Sec. 6.6.

The next proposition gives an analogous construct for disjunctive parametric heterogeneous verification.

Proposition 5.5. Given parameterized system model M_0 and specification S_0 with a behavior formalism B_0 and the pair of constraints C_0^M, C_0^S over the system-level parameters P_0^M, P_0^S, a set of parameterized models M_i and specifications S_i with corresponding behavior formalisms B_i, behavior relations $R_i \subseteq B_0 \times B_i$ and pairs of constraints C_i^M, C_i^S over parameters P_i^M, P_i^S for $i = 1, \ldots, n$, if

i. constraints (C_i^M, C_i^S) are external-constraint consistent,

ii. models M_i form a parametric cover for M_0,

iii. specifications S_i each parametrically imply S_0 and

iv. $C_i^M, M_i \models^{B_i} C_i^S, S_i$

then $C_0^M, M_0 \models^{B_0} C_0^S, S_0$.

Proof. From the definition of parametric coverage, there exists a partition $\{B_0^1, \ldots, B_0^n\}$ of

$[\![C_0^M, M_0]\!]^{B_0}$ s.t.

$$
\begin{aligned}
[\![C_0^M, M_0]\!]^{B_0} &\subseteq \bigcup_i R_i^{-1}([\![E_i^M, M_i]\!]^{B_i}) \\
\text{(Def. 5.6, monotonicity)} &\subseteq \bigcup_i R_i^{-1}([\![C_i^M, M_i]\!]^{B_i}) \\
(C_i^M, M_i \models^{B_i} C_i^S, S_i) &\subseteq \bigcup_i R_i^{-1}([\![C_i^S, S_i]\!]^{B_i}) \\
\text{(Def. 5.6, monotonicity)} &\subseteq \bigcup_i R_i^{-1}([\![E_i^S, S_i]\!]^{B_i}) \\
\text{(Def. 5.4)} &\subseteq [\![C_0^S, S_0]\!]^{B_0}
\end{aligned}
$$

Therefore, $C_0^M, M_0 \models^{B_0} C_0^S, S_0$. □

This proposition presents the parametric counterpart of the disjunctive heterogeneous analysis construct from Prop. 5.3. It uses the parametric definitions of coverage (Def. 4.5), entailment (Eq. (3.2)) and implication (Def. 5.4). The external-constraint consistency and monotonicity provide the remaining pieces for the conjunctive parametric verification to work out.

5.5 Hierarchical Heterogeneous Verification

We note that the conjunctive and disjunctive analysis constructs can be nested arbitrarily. For example, the j^{th} conjunctive verification subtask $M_j \models^{B_j} S_j$ can be broken down disjunctively into its subtasks $M_{ji} \models^{B_{ji}} S_{ji}$ by creating new models that cover M_j and specifications that imply S_j. Thus, using the nesting of conjunctive and disjunctive constructs, any arbitrary propositional logical breakdown of a system verification task can be

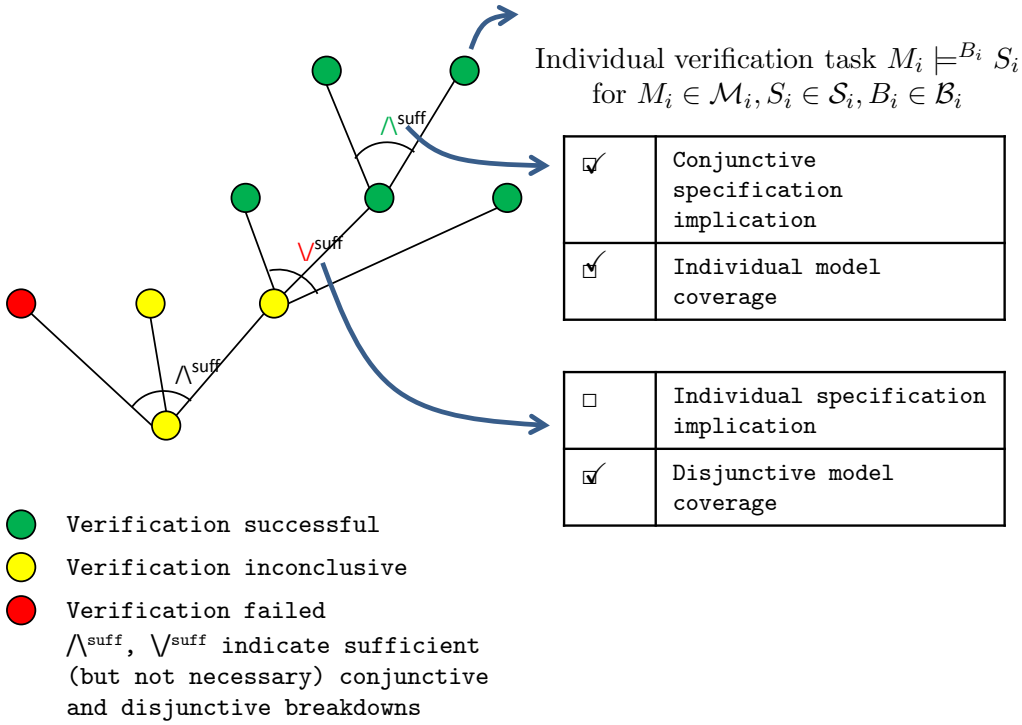


Figure 5.6: Heterogeneous verification as a tree representation.

achieved. This is illustrated using a case study in Sec. 6.2.

In its most basic form, a heterogeneous verification can be viewed as a tree, as illustrated in Fig. 5.6. Similar tree structures for organizing information are used in tools such as SVM [7] for requirements management and in fault-tree analysis tools such as PLFaultCAT [33] and Galileo [85] for analyzing root causes of faults. In a heterogeneous verification tree, each node is a verification activity and can be successful, inconclusive or failed. The root node is the system verification activity, with subsequent children nodes representing the verification activities invoked to reason about the parent verification activity. The success of the verification activity of a parent node can be inferred from the successful verification activities of its children nodes plus the correctness of the conjunctive

or disjunctive decomposition. If either the verification of children nodes or the correctness of decomposition cannot be established, the analysis result of the node is inconclusive (and not failed), because these conditions are sufficient, but not necessary. Associated with each node are the details of the verification activity, namely the model, its specification, a list of parameters, the constraints on parameters in the model and the specification, and a behavior relation indicating how the behaviors of children relate to the behaviors of the parent.

5.6 Summary

This chapter develops a framework based on behavior relations for enabling heterogeneous verification. Behavior relations provide with the semantic associations across heterogeneous behavior domains, which allow us to extend the usual notions of abstraction and implication to heterogeneous domains. Multiple abstractions can be used together towards the verification with an underlying more-detailed model in two kinds of ways — one where models are individually abstractions and specifications on the models together imply the underlying specification, and one where models for different operating regimes together abstract the underlying system model and specifications individually imply the underlying specification. When the model can switch between different operating regimes over time, initial conditions used for verification need to correctly overapproximate the range of initial conditions that are physically possible due to the mode switching.

Chapter 6

Case Study

In this chapter, we demonstrate the practical applicability of the theoretical machinery developed in this thesis. We use an example from the automotive CPS domain that is both heterogeneous and safety-critical.

Every year, police reported automotive crashes in the United States amount to $ 300 billion in comprehensive costs and 43,000 fatalities. Out of these, intersection-area crashes amount to $97 billion in comprehensive costs and 9,500 fatalities [25]. Cooperative intersection collision avoidance systems (CICAS) is a government-industry initiative to instrument intersections and make use of these smart intersections to cooperate together with smart vehicles to eliminate intersection-related crashes [1].

Within the CICAS umbrella, three types of intersection collisions are considered. CICAS for violations (CICAS-V) looks at collisions caused due to stop-sign and signal-controlled intersection violations. CICAS for signalized left turn assist (CICAS-SLTA) focuses on signalized left turns at intersections with no protected left turn, where a driver

turning left needs to make a judgment about the safety of gaps in the oncoming opposing traffic. CICAS for stop-sign assist (CICAS-SSA) focuses on rural highways with stop-sign controlled intersections where a driver crossing the highways needs to make a judgment about the safety of the gaps in the oncoming lateral traffic [1].

The instrumented intersection consists of a traffic signal controllers capable of exporting signal phase and timing in formation, a local global positioning system (GPS), and road side equipment (RSE) that includes computing, memory, and dedicated short range communication (DSRC) radio. The vehicle portion of the systems includes on-board equipment for computing and DSRC radio connected to the vehicle controller area network (CAN), global positioning, and the driver-vehicle interface (DVI). The instrumented intersection sends the signal phase and timing, positioning corrections, and positions and velocities of the oncoming vehicles as applicable to either the RSE or to a DSRC-equipped vehicle. The computation engine at RSE or the vehicle uses the available information to analyze the safety of a given gap and informs that to the driver via DVI or a roadside dynamic message sign [65].

MBD of CICAS presents the challenges we aim to address in this thesis. The systems are safety-critical due to the very nature of the application, which warrants formal verification. The systems are also heterogeneous due to the diversity of the constituent elements such as sensing of positions and velocities of vehicles, communication of the signal phase and timing information and sensor readings to a computer, software to compute safe gaps either on a computation element based on the physical dynamics of the vehicles and speed limits. There is no good unified formalism for modeling all aspects of such complex het-

erogeneous system, yet one would like to formally verify the correctness of such a system. Therefore, these systems are good applications for illustrating the practical implications of the theoretical developments in this thesis. In this chapter, we use CICAS-SSA as a representative system from CICAS and develop its hierarchical heterogeneous verification.

6.1 Cooperative Intersection Collision Avoidance System for Stop-Sign Assist

Rural highways in the United States often have intersections where minor road traffic is allowed to cross the highway traffic. Drivers on minor roads at these stop-sign controlled intersections have to determine when the gap in the cross traffic is sufficient for them to drive across the highway or make a turn to merge into the highway traffic safely. A number of factors lead to errors in human judgment about the safety of oncoming gaps, such as lack of clear visibility due to the intersection geometry or inclement weather, error in judgment about the speed of oncoming vehicles, and inaccurate estimates about how long it will take to drive through the intersection or make a turn onto the highway and blend safely with the cross traffic. Accidents at such intersections are often fatal because of the side-on collisions at highway speeds. Over 60% of the fatal crashes occur in rural stop-sign controlled highway intersections . Fatalities and injuries resulting from stop-sign related crashes cost approximately $28 billion annually [67].

CICAS-SSA is a research initiative by the US Department of Transportation Federal Highway Administration, Minnesota Department of Transportation and University of Min-

nesota's Intelligent Transportation Systems (ITS) Institute [2]. The goal of the program is to instrument the rural stop-sign controlled intersections to help human driver judge the safety of oncoming traffic gaps by: sensing the positions and velocities of the oncoming traffic; communicating these readings to a computer; computing safety of the oncoming gap based on the intersection geometry, type of the vehicle wanting to cross the traffic, and speed limits; and displaying the (un)safety of the gap to the driver on a dynamic message sign.

Figure 6.1 shows a schematic of a rural highway called the *major road* with a *minor road* crossing it at a stop-sign controlled intersection. The major road is monitored for the positions and velocities of the oncoming vehicles and the minor road is monitored for occupancy and vehicle class. The vehicle at the stop-sign being served by CICAS-SSA is called the *subject vehicle* (SV). The nearest oncoming vehicle is called the *principal other vehicle* (POV), where 'nearest' is defined as the vehicle in the oncoming traffic with the shortest *time-to-intersection* as determined from the positions and approach velocities of the vehicles in the oncoming traffic. In the example in Fig. 6.1, the two-way highway is separated by a median that is wide enough for a vehicle to stop between the traffic in each direction, so the median is monitored for occupancy.

Following the terminology used in the CICAS-SSA literature, the distance between any two vehicles approaching the intersection is called a *gap*, while that between a POV and the intersection center line is called a *lag*. As oncoming cars enter and cross the intersection, the downstream gaps become lags, and the safety-judgment decision of whether or not to enter the intersection can be made based on the lags between the nearest oncoming vehicles

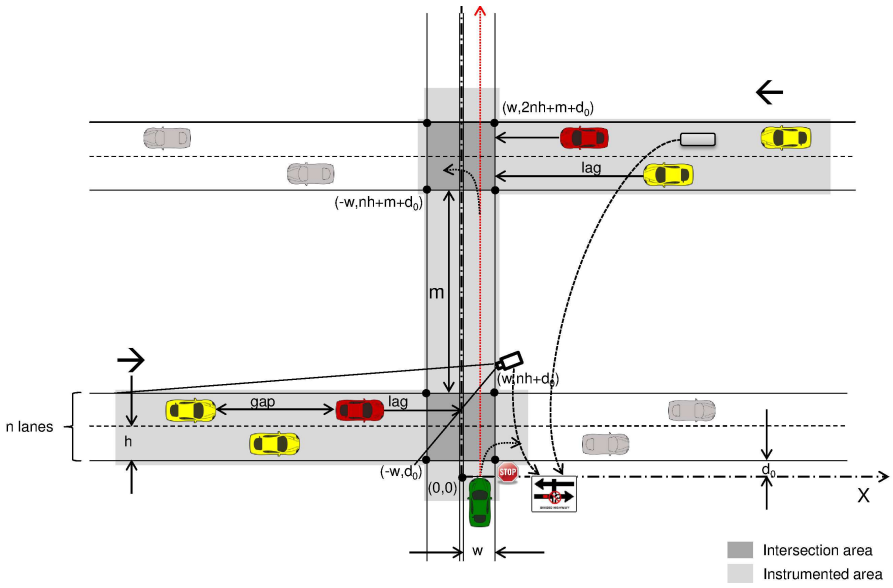


Figure 6.1: A pictorial sketch of CICAS-SSA.

in each lane and the intersection.

CICAS-SSA being implemented in prototypes warns drivers when it is unsafe to enter the intersection and telling them to proceed with caution otherwise [45]. This simple warning scheme is based on empirical data, based on the smallest size of gap in seconds that 80 percentile of drivers would reject. Rather than using this simple approach of alerting or warning the driver, we consider the scenario that a more conservative advice is given to the driver, but one that is guaranteed to be safe, i.e., if the driver does follow the instructions, there cannot be a collision. The particular strategy we model in this case study is shown in Fig. 6.2. The SV modeled is allowed to (but doesn't have to) enter the intersection only if all the oncoming vehicles are far enough away (beyond a distance l) from the intersection to allow the SV to pass completely through the intersection before the POV has arrived at the intersection. Otherwise the SV has to remain stopped. The driver model for SV assumes that (s)he responds within a finite duration of time and accelerating

with a minimum value once starting to drive.

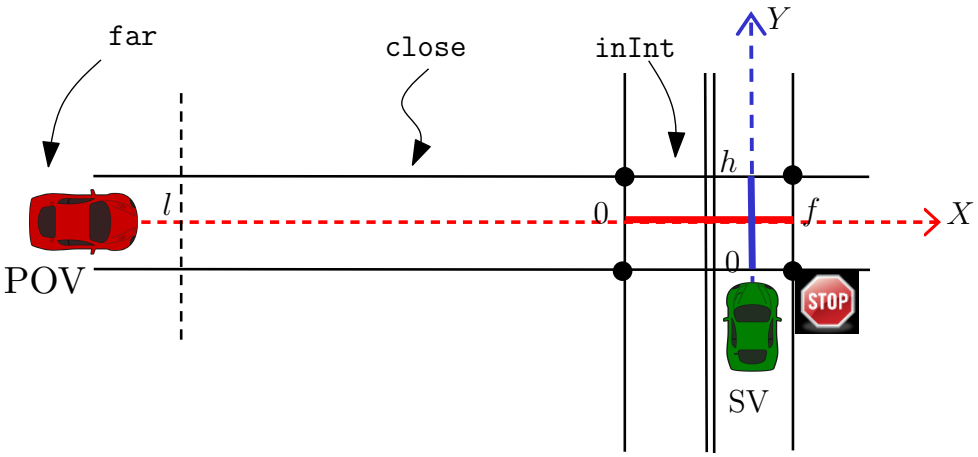


Figure 6.2: Safety-decision schematic for CICAS-SSA.

6.2 Hierarchical Heterogeneous Verification Tree for CICAS-SSA

The verification objective for the CICAS-SSA design described above is to ensure that if a driver follows the SV driver follows the system's advice, there is no collision. We use the hierarchical heterogeneous tree representation from Sec. 5.5 to manage several levels of abstraction to break this problem down into heterogeneous models and their analysis using nested conjunctive and disjunctive verification constructs from Sec. 5.3.1, Sec. 5.3.2 and Sec. 5.4.

Figure 6.3 depicts the heterogeneous verification tree for for the CICAS-SSA design discussed above. Each node in the verification tree is a verification activity, with the root identifying the system-level verification. We enumerate each level of abstraction as

the distance from the root node in terms of the number of links, i.e., Levels 0 (for root) through 6 (for leaf nodes). We refer to each node with the index ij where i is the level of abstraction and j is a number read from left to right starting from 1 at a given level of abstraction. Conjunctive breakdowns are depicted with an arcs labeled $\wedge$, and disjunctive breakdowns and disjunctive coverage for mode switching are depicted with arcs labeled $\vee$ and $\vee^*$. We give a high-level description of the verification tree first, and selected pieces of this tree in detail later to illustrate various aspects of the theory.

The verification for each node can be concluded by the verification of its children one level of abstraction up, given a correct method of breakdown used – conjunctive, disjunctive or disjunctive with switching. At each level of abstraction, depending on the nature of the breakdown, one has to create models, define new specifications for them, construct behavior relations and verify model abstractions or coverage and conjunctive or individual specification implication. Abstraction can be carried out compositionally using behavior abstraction functions whenever possible. The actual verification activity is carried out at the leaf nodes using some analysis techniques and tools.

The verification objective at the root node, Node 01, can be established as "given the uncertainties in the measurements from the sensing subsystem, delays in the communication subsystem, computation time, and driver response time, there is never a collision if the SV driver follows the system's advice." At abstraction level 1, different models of communication, computation, sensing and driver behavior are used to obtain bounds on delays and errors which are used along with a formal verification model to establish SV and another car is never in the intersection at the same time. At level 2, the formal verifica-

tion model is disjunctively covered by the three choices the SV has, namely going straight, turning right and turning left. At level 3, the models for each of these cases is conjunctively abstracted by simpler models that consider only one lane at a time. At level 4, the single lane-multi-vehicle models are covered with mode-switching by models with a single POV that either starts safe or starts unsafe. For the case when the POV starts safe and the SV has a chance to enter the intersection, at level 5 (leaf nodes) we have simple models that capture the dynamics of the POV, the dynamics of the SV and a discrete protocol model. The relative times of POV to get to and SV to exit the intersection along with the correct order ensures conjunctive heterogeneous verification. The leaf nodes are actually verified, without constructing any further abstractions.

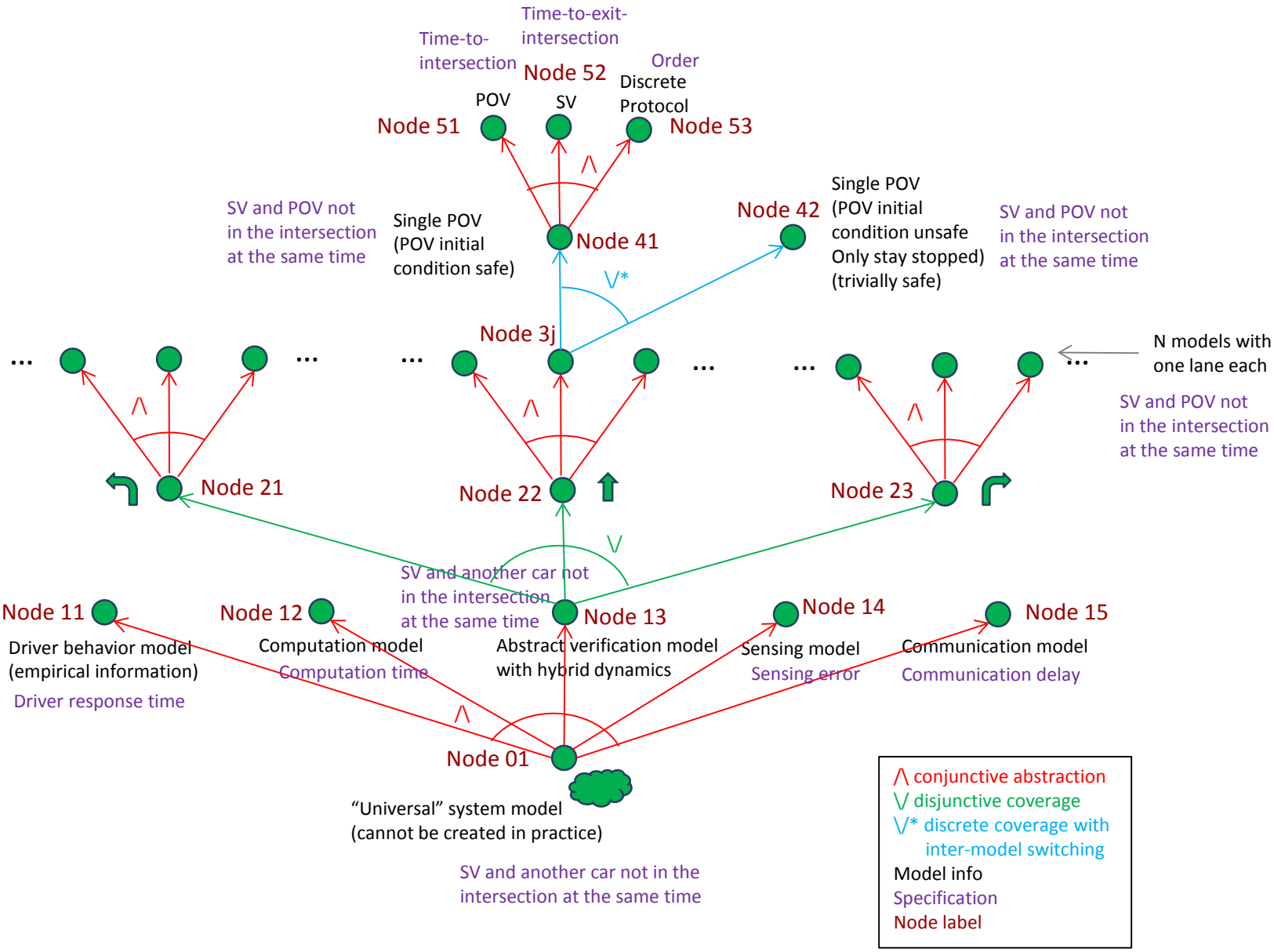


Figure 6.3: CICAS-SSA hierarchical heterogeneous verification tree.

6.3 Disjunctive Heterogeneous Verification

In this section we illustrate the theoretical concepts of disjunctive heterogeneous verification developed in Sec. 5.3.2. Disjunctive heterogeneous verification involves creating heterogeneous abstractions that cover different subsets of behavior sets of an underlying model. The specifications that are checked against these models need to each individually imply the original specification. A special case of coverage can be used for mode-switching systems while verifying temporally invariant specifications. Next we describe two places within the CICAS-SSA tree in Fig. 6.3 where disjunctive heterogeneous verification can be used.

6.3.1 Model Coverage Using Behavior Relations

We consider the verification problem at Node 13 in Fig. 6.3 and illustrate the concepts of coverage using behavior relations, specification implication using behavior relations and their use for disjunctive heterogeneous verification.

Consider the verification model M_{13} at Node 13 as shown in Fig. 6.4. The model is made up of two hybrid automata components **Major Road** and **SV**. The **Major Road** component models the dynamics of the oncoming vehicles. The vehicle dynamics are modeled by differential inclusions representing ranges of possible velocities given highway speed limits. The vector-valued variable x in Fig. 6.4 represents the positions of the oncoming cars from the intersection, which are negative in the frame of reference for the intersection.

In real intersections, the oncoming vehicles approach the intersection one after another. As they enter the intersection and cross, new vehicles become the ones of interest. This

phenomenon can be thought of as a loop that initializes the oncoming vehicles to their positions, and runs until some vehicle reaches the intersection, and after that vehicle has crossed the intersection, the system is re-initialized with the positions of the new set of vehicles, which again runs until the next vehicle reaches the intersection. The model captures one such instance that is useful over and over again. This instance allows the evolution of the system to continue until some car reaches the intersection, represented by the invariant $x \leq 0$ in the **Major Road** component. It also captures the worst-case set of initial conditions for the oncoming cars: they can be anywhere in the instrumented area, denoted by the range $[-420, 0]$ for the initial conditions. Proving the infinite loop, given these individual instances, would be necessary to infer the correctness of the real systems.

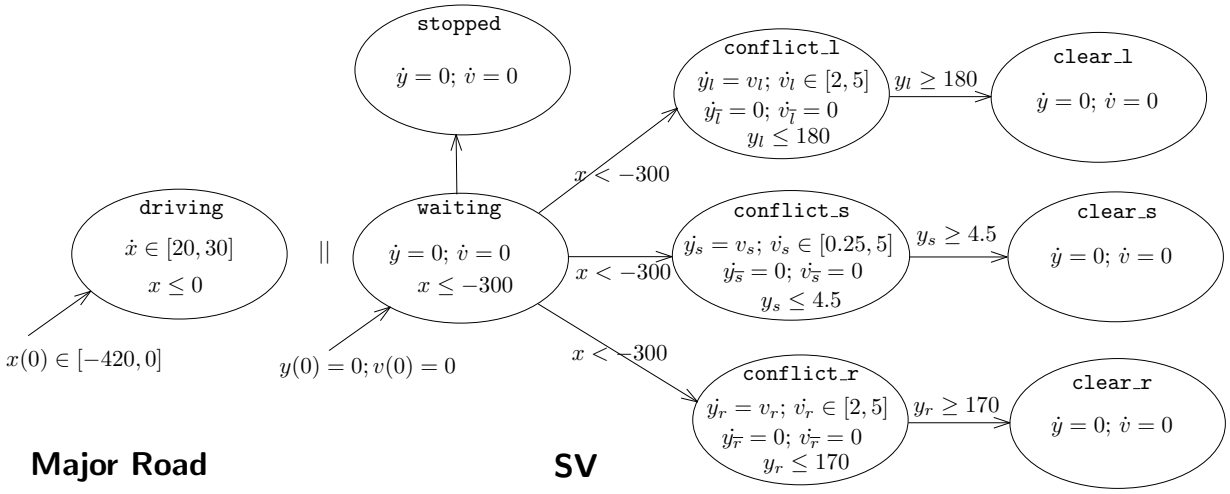


Figure 6.4: Verification model M_{13}.

The decision strategy modeled for the SV is that if all the oncoming vehicles on the major road are beyond an imaginary marker at position $l = -300$, the SV is permitted to start driving, but it doesn't have to. When some car crosses l, the SV has to stay stopped, forced by the invariant in `waiting`. Whenever permitted, whether the SV decides to go

straight or turn left or right is represented as a nondeterministic choice; however once it has committed to one, it isn't allowed to change its mind. The position and velocities of the SV along straight, left and right directions are y_s, y_l, y_r and v_s, v_l, v_r respectively. For brevity of notation in Fig. 6.4, y and v mean $\{y_s, y_l, y_r\}$ and $\{v_s, v_l, v_r\}$, respectively; $y_{\bar{s}}$, $y_{\bar{l}}$, $y_{\bar{r}}$ mean $\{y_l, y_r\}$, $\{y_s, y_r\}$, $\{y_s, y_l\}$, respectively; and $v_{\bar{s}}$, $v_{\bar{l}}$, $v_{\bar{r}}$ mean $\{v_l, v_r\}$, $\{v_s, v_r\}$, $\{v_s, v_l\}$, respectively.

The evolution of the model M_{13} stops when the SV clears the conflict regions or when the some major road vehicle enters the intersection. By the time the major road vehicle enters the intersection, if the SV is still in the conflict zone, there is a safety violation (a potential collision). Alternatively, if the SV has cleared the conflict zone or hasn't entered it, there is no safety violation. The objective is to guarantee collision freedom for this particular strategy. The collision-freedom specification S_{13} can be defined by the temporal logic formula $\Box \neg ((x == 0 \wedge 0 < y_s < 4.5) \vee (x == 0 \wedge 0 < y_r < 170)) \vee (x == 0 \wedge 0 < y_l < 180))$, where numbers 4.5, 170 and 180 are chosen based on a typical highway intersection geometry.

We create three models shown in Fig. 6.5, 6.6 and 6.7, for the cases where SV is only allowed to turn left, go straight and turn right.

The behavior domain of M_{13} (i.e., B_{13}) is the set of all $MN + 6$ dimensional hybrid traces, where N is the number of major road lanes with M being the max number of vehicles that can fit within the instrumented area per lane, yielding MN as the dimension for x plus 6 from SV. The dimensionality of the SV model is reduced when we only consider one direction at a time, and so the domains B_{21}, B_{22} and B_{23} are each sets of all $MN + 2$

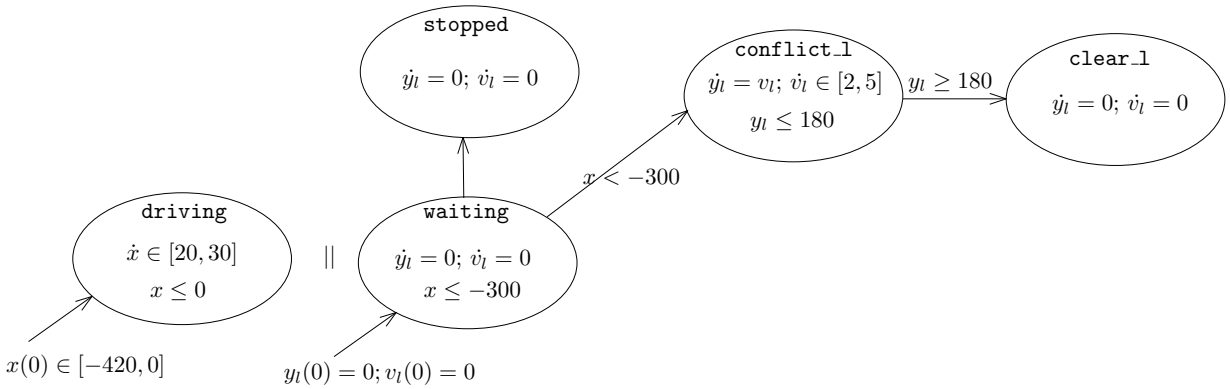


Figure 6.5: Verification model M_{21}.

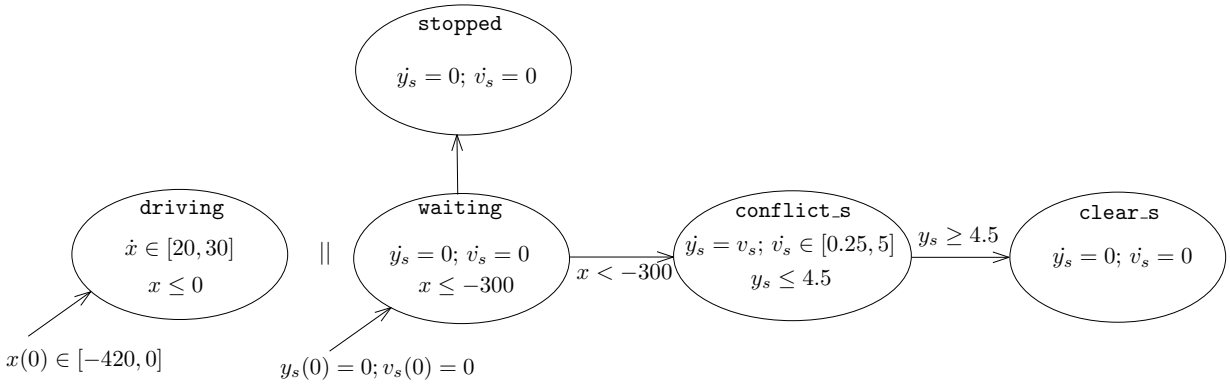


Figure 6.6: Verification model M_{22}.

dimensional hybrid traces. The behavior relations for this breakdown are as follows:

- $R_{21} : \{(b_{13}, b_{21}) | b_{13} \downarrow_{y_s,y_r,v_s,v_r} == \bar{0}$ and $b_0 \downarrow_{x,y_l,v_l} == b_{21}\}$
- $R_{22} : \{(b_{13}, b_{22}) | b_{13} \downarrow_{y_l,y_r,v_l,v_r} == \bar{0}$ and $b_0 \downarrow_{x,y_s,v_s} == b_{22}\}$
- $R_{23} : \{(b_{13}, b_{23}) | b_{13} \downarrow_{y_l,y_s,v_l,v_s} == \bar{0}$ and $b_0 \downarrow_{x,y_r,v_r} == b_{23}\}$

where $\downarrow_{<<list>>}$ represents the projection onto the list of variables $<< list >>$ and $\bar{0}$ represents vector traces of zeros over all time .

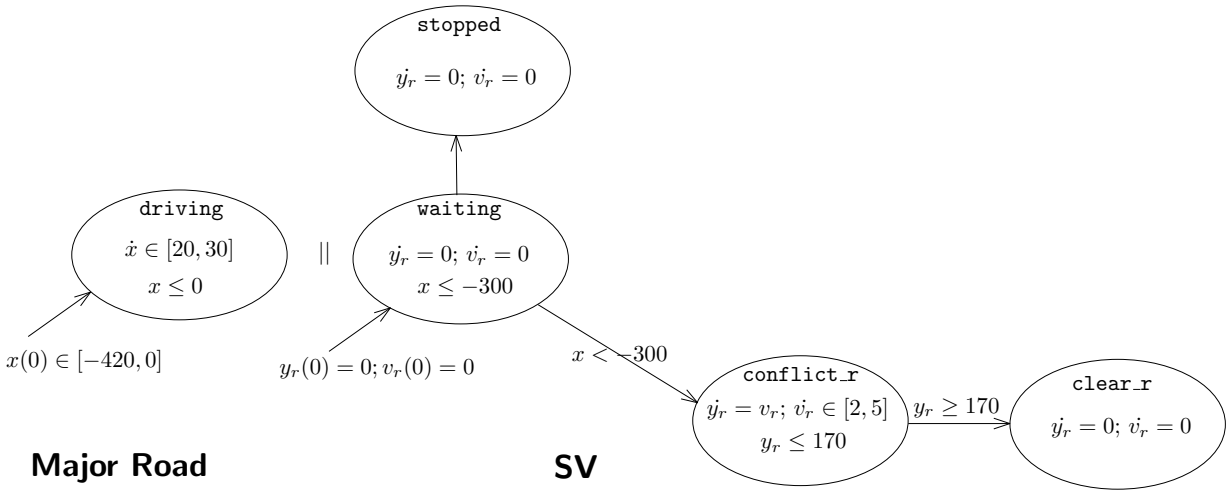


Figure 6.7: Verification model M_{23}.

The specifications to be checked for the models are

- $S_{21} : \Box \neg (x == 0 \wedge 0 < y_l < 180)$,
- $S_{22} : \Box \neg (x == 0 \wedge 0 < y_s < 4.5)$ and
- $S_{23} : \Box \neg (x == 0 \wedge 0 < y_r < 170)$.

We have heterogeneous implication $S_{21} \Rightarrow^{R_{21}} S_{13}$ because $R_{21}^{-1}(\ [\![S_{21}]\!]^{B_{21}})$ forces that y_l be conflict-free and y_s, y_r be 0, which implies that y_l, y_s, y_r are conflict-free. Similarly, we have $S_{22} \Rightarrow^{R_{22}} S_{13}$ and $S_{23} \Rightarrow^{R_{23}} S_{13}$. Further, we note that in every behavior of M_{13}, has only $\{y_l, v_l\}$ or $\{y_s, v_s\}$ or $\{y_r, v_r\}$ nonzero and rest SV variables zero. Each of these possibilities is covered by one model. Therefore, from Prop. 5.3, if $M_{2i} \models^{B_{2i}} S_{2i}$, we can conclude $M_{13} \models^{B_{13}} S_{13}$.

To summarize, we use the concepts of model coverage using behavior relations, specification implication using behavior relations and demonstrated their use towards the disjunctive heterogeneous verification problem at Node 13 in Fig. 6.3. This disjunctive coverage reduces the dimensionality of the **SV** variables and breaks the verification problem down

into individual cases for SV turning left, going straight or turning right.

6.3.2 Model Coverage for Mode Switching

Now we illustrate the notion of model coverage for mode switching from Sec. 4.1 and demonstrate its use for verifying temporally invariant specifications in isolation for each individual mode.

Consider the analysis task at Node $3j$ of establishing that the SV and another car are never in the intersection at the same time, for models that include a single lane but several vehicles. As noted earlier, the safety decision is made based only on the nearest vehicle, which is the POV. Once the closest oncoming vehicle crosses the intersection, it "falls off" the instrumented area and the next oncoming vehicle becomes the new POV. This behavior can be covered by two models with inter-model switching for the following two cases.

Figure 6.8 shows the mode-switching schematic of a car falling off the instrumentation area and a new one becoming the new POV. This event is modeled as the discrete jump and a reset of the continuous variable x, which always maintains the position of the POV. As oncoming POVs cross the intersection, the new ones can be either initially safe (modeled by M_{41}) or already unsafe (modeled by M_{42}). Since the specification of establishing that the SV and any other car is never in the intersection at the same time is a temporally invariant requirement, we can reason about the requirement for the two modes independent of how many times and in what order the system switches between the two modes. Note that we have disjunctive coverage with inter-model switching so long as initial conditions are overapproximative, and here they are.

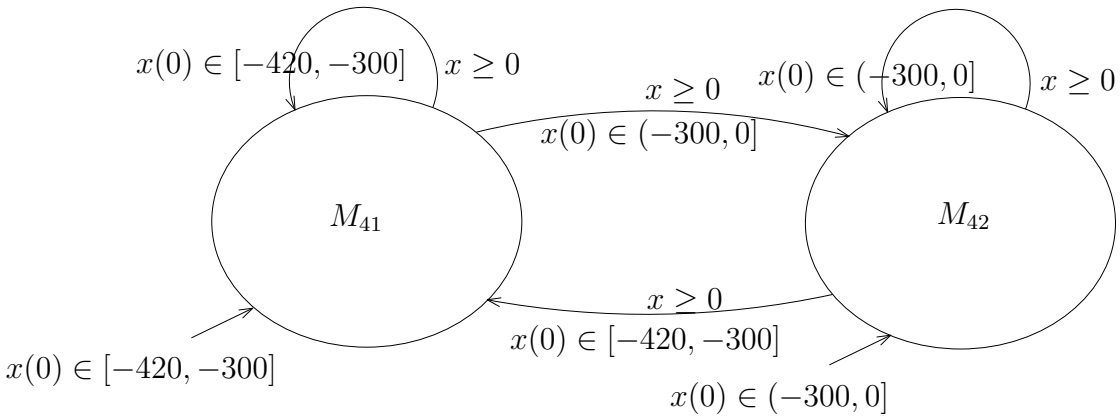


Figure 6.8: Inter-model switching that covers M_{3j} .

In case of Node 42, the POV is already too close for the SV to enter the intersection safely, so the SV can only stay stopped, so it is trivially safe. For Node 41 when the POV is initially at a safe distance, the SV can decide to enter the intersection if it chooses to. This is modeled in model M_{41} as shown in Fig. 6.9. In the first case, whenever the current POV crosses the intersection and a new vehicle becomes the POV, the new POV is still far off, i.e., beyond the reference marker. In this case, the SV can possibly start driving, but doesn't have to. This case is considered in Node 41. In the other case, it is already close enough, and the SV cannot start driving and has to stay stopped. This is the trivial case considered in Node 42.

To summarize, we have used the notion of coverage with mode switching to construct two different models for two modes that represent the POVs approaching the intersection in CICAS-SSA. No matter how many cars there are on the highway, each oncoming car when it becomes the POV either starts far off or too close for SV to safely enter the intersection. This lets us simplify the dimension of the single-lane multi-vehicle Major Road models from M at Nodes 3j to one for single-lane single-POV models at Nodes 41 and 42. Because

we are interested in safety over all time, the different SV actions for the two models can be analyzed independently for safety over all time while the system is in those modes.

6.4 Conjunctive Heterogeneous Verification

In this section, we illustrate the notion of conjunctive heterogeneous verification from Sec. 5.3.1. Conjunctive heterogeneous verification involves creation of models that form abstractions of the underlying model and checking specifications against these models that conjunctively imply the underlying specification. The heterogeneity in modeling and specification formalisms can be addressed by either defining the semantics in a common domain or using behavior relations as illustrated next.

6.4.1 Conjunctive Abstraction in a Common Semantic Domain

We illustrate the use of heterogeneous abstraction with the modeling heterogeneity resolved by defining the model semantics in a common system behavior domain. This lets us use compare the semantic interpretations of heterogeneous models and specifications in the same behavior domain. This method is a good choice at Node 01 in which we model the system-level verification task. The semantics of the various abstractions constructed are defined in terms of this system behavior domain.

At Level 1, we have different aspects of the system design being modeled and analyzed independently for the respective concerns. For example, Node 11 captures the driver behavior with an objective of getting a conservative estimate on the response time. Node 12

captures a computational model and the objective is to do some worst-case execution time analysis to determine the timing. Node 14 represents the analysis of the sensing subsystem, with the objective of making sure that the sensing error is always bounded within a given range of positions and velocities. Node 15 represents a communication model used to find out a bound on the communication delay from sensing subsystem to the computation subsystem. Node 13 is a formal model with hybrid dynamics to be used to formally establish that "`the SV and any other vehicle are not in the intersection at the same time`" for the given computation time, communication time, sensing errors and driver response time.

The semantics of these models can be defined in terms of a common system behavior domain. Note that all the models are abstractions of the underlying system because they restrict only the behaviors in the analysis aspect while allowing everything in the rest of the aspects of the system. The specifications conjunctively imply specification S_{01} because S_{15} is the same as S_{01}. The rest of the specifications are indirectly used in qualifying the behaviors of interest in the verification model. The measurement error is overapproximated in the nondeterminism in the verification model M_{15}, while the driver response, computation and communication time delay bounds τ_d, τ_c, τ_{cc} are accounted for in the difference between time-to-intersection and time-to-exit-intersection at the leaf nodes. Note that without the support for auxiliary constraints over parameters, we would not be able to model these dependencies.

In summary, modeling and specification heterogeneity can be addressed by defining their semantic interpretations in a common behavior domain. This can be done relatively

easily when typically each model represents different aspects of the same underlying system. There are other places in the verification tree in Fig. 6.3 where conjunctive analysis constructs can be used, namely Nodes 2i at Level 2, and Node 41 at Level 4. One can use common behavior domains to resolve interdependencies, but we use different behavior domains and use behavior relations to address the heterogeneity instead as illustrated next.

6.4.2 Conjunctive Abstraction via Behavior Relations

We now illustrate the notion of conjunctive heterogeneous verification using behavior relations when model and specification semantics are defined in terms of different behavior domains.

Consider the verification task of showing $M_{41} \models^{B_{41}} S_{41}$ at Node 41 in Fig. 6.3. The model M_{41} is shown in Fig. 6.9. We break down this task conjunctively by creating three models M_{5i} as shown in Fig. 6.10 and constructing corresponding specifications S_{5i}, $i = 1, 2, 3$. M_{51} models the behaviors of the POV, and is exactly the same as the POV automaton in M_{41}. M_{52} models the behavior of the SV only while it is in the conflict zone and has the same dynamics as that of the `conflict_s` location of M_{41}. M_{53} is a discrete model consisting of two elements. The component POV is a created by partitioning the component POV of M_1 into discrete states `far`,`close`, and `inInt` using predicates $x \leq -300$, $-300 \leq x \leq 0$, and $0 \leq x$. The second component SV is merely a discrete control graph of the hybrid automaton model for SV in M_1. The only synchronized pair of transitions is ($\texttt{far} \xrightarrow{\beta_1} \texttt{close}$) and ($\texttt{waiting} \xrightarrow{\beta_1} \texttt{stopped}$).

The behavior domain B_{41} is the set of 3-d hybrid trajectories in variables x, y_s and v_s.

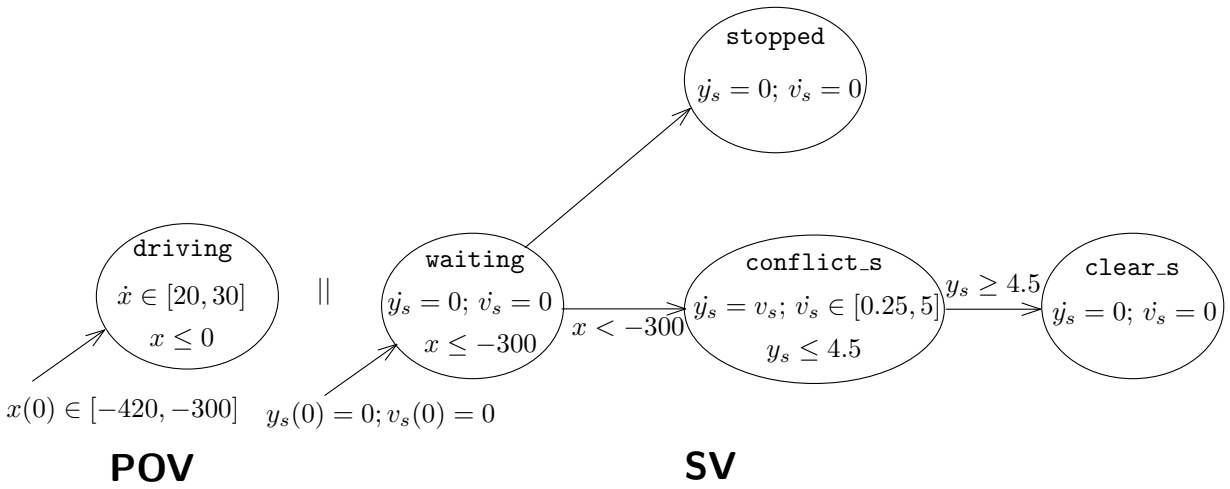
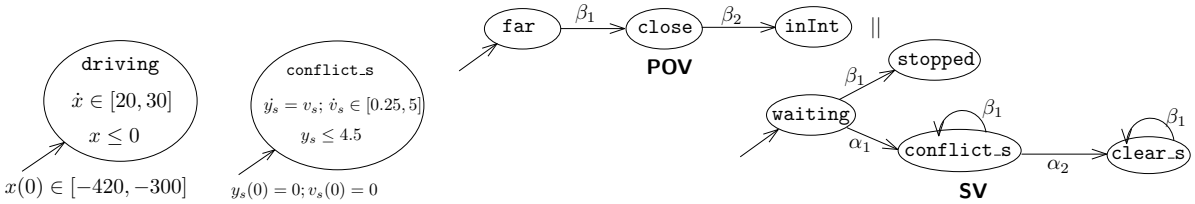


Figure 6.9: A hybrid model M_{41} for SV going only straight if safe.



(a) Model M_{51} (b) Model M_{52} (c) Model M_{53}

Figure 6.10: Heterogeneous abstractions M_{5i} of M_{41}.

The behavior domain B_{51} and B_{52} are 1-d and 2-d continuous trajectories in variables x and y_s, v_s respectively. The behavior domain B_{53} is $\{\alpha_1, \alpha_2, \beta_1, \beta_2\}^*$. The behavior relations are

- $R_{51} : \{(b_{41}, b_{51}) | b_{51} == b_1 \downarrow_x \}$,
- $R_{52} : \{(b_{41}, b_{52}) | b_{52} == s_{41} \downarrow_{y_s, v_s}$ where s_{41} is b_{41} restricted to the discrete location `(driving,conflict_s)`$\}$ and
- $R_{53} : \{(b_{41}, b_{13}) | b_{41}$ is a hybrid trajectory that visits the discrete locations corresponding to ones in b_{53} in that order $\}$.

For these behavior relations, we first note that $M_{41} \sqsubseteq^{R_{5i}} M_{5i}$ because neither of the

models M_{5i} is more restrictive than M_{41}. The specifications for the three models are

- $S_{51} : \Box\ (x == -300 \Rightarrow \Box_9\ x < 0)$,
- $S_{52} : \Box\ (\Diamond_8\ y_s \geq 4.5)$ and
- $S_{53} : \Box\ ((\phi_1 \wedge \neg\phi_2) \rightarrow \neg(\Diamond\phi_2))$, where ϕ_1 is the predicate "`POV is close`" satisfied in states (`close`,$\cdot$) and (`inInt`,$\cdot$); and ϕ_2 is the predicate "`SV is driving`" satisfied in states ($\cdot$,`conflict_s`).

The behaviors effectively allowed in B_{41} by the specifications S_{5i} are as follows:

- $R_{51}^{-1}([\![S_{51}]\!])$: system behaviors where POV takes at least 9 seconds to get from $l = -300$ to the intersection.
- $R_{52}^{-1}([\![S_{52}]\!])$: system behaviors where SV clears the intersection within 8 seconds of starting to drive.
- $R_{53}^{-1}([\![S_{53}]\!])$: system behaviors where SV does not start driving after POV crosses l.

There can only be two cases:

1. The SV has already started driving before the POV crosses l and is in the intersection: in this case, from $R_{51}^{-1}([\![S_{51}]\!])$ and $R_{52}^{-1}([\![S_{52}]\!])$ together, the SV will clear the intersection in at most 8 seconds and the POV won't get to the intersection in at least 9 seconds, OR
2. The SV hasn't started driving when the POV crosses l: in this case, from $R_{53}^{-1}([\![S_{53}]\!])$, the SV cannot start driving anymore.

Therefore, from all the specifications put together, the two cars can't be in the intersection at the same time, which implies S_{41}, i.e., we have conjunctive heterogeneous implication.

$M_{51} \models^{B_{51}} S_{51}$ can be shown by algebraic computations: for the fastest velocity (30m/s) it takes 10s to travel 300m. $M_{52} \models^{B_{52}} S_{52}$ can be shown by Newton's laws of motion: the longest time needed to cross 4.5m with initial velocity 0 and minimum acceleration 0.25m/s^2 is $\sqrt{\frac{2*4.5}{0.25}} = 6$ seconds. $M_{53} \models^{B_{53}} S_{53}$ can be shown by using Labeled Transition System Analyzer (LTSA).

Because of all the conditions for the correct conjunctive heterogeneous verification are satisfied, from Prop. 5.2, we can conclude $M_{41} \models^{B_{41}} S_{41}$.

To summarize, we have used conjunctive heterogeneous verification using behavior relations when model and specification semantics are defined in different behavior domains. In contrast with the method of defining semantics in a common domain, this approach lets us define individual semantics in domains of choice, which can be helpful for leaf nodes where there is no further breakdown and verification activities directly take place.

Conjunctive heterogeneous abstraction can also be used at Nodes 2i, where each model M_{2i} with multi-lane multi-vehicle **Major Road** component models can be conjunctively abstracted by N single-lane multi-vehicle models with M vehicles, which reduces the **Major Road** dimension down to N instances of M each. The definitions of the conflict zones in the straight, left and right directions are different for different lanes when considered individually, since the lanes are geographically at different coordinates. One can address these different values using a conservative worst-case approach by using conflict-zone definitions that include all the actual individual conflict zones for each of the lanes. Alternatively, we can also treat the conflict zone boundaries as parameters and use parameter interdependency framework introduced in Sec. 6.6 for doing analysis for different valuations of these

parameters corresponding to the different lanes.

6.5 Compositional Heterogeneous Abstraction

In this section, we illustrate the notions of compositional heterogeneous abstraction developed in Sec. 4.2, particularly using distributed development with component abstractions established in local behavior domains. Compositional heterogeneous abstraction involves defining the local semantics of component models, creation of behavior abstraction functions and establishing abstraction via them, and ensuring that consistency conditions between the abstraction functions are met such that agreeing globalized system-level abstraction function can be constructed and used.

Consider the problem of establishing abstraction between models M_{41} and M_{53} as shown in Fig. 6.9 and 6.10(c). We establish heterogeneous abstraction between these two models compositionally in a distributed manner. We call the POV and SV components within models M_{41} and M_{53} as component models P_0, Q_0 and P_1, Q_1 respectively. We use two different kinds of abstraction functions for two components – one using state-space partitioning and another by retaining the discrete transition graph by projecting away all the continuous dynamics.

6.5.1 Heterogeneous Abstraction for POV

The local behavior domains for the POV models P_0 and P_1 are B_0^{POV}: 1-d hybrid traces, i.e., evolution of the hybrid state $h^{POV} := (l^{POV}, x)$ over time, with $l^{POV} \in \mathcal{L}^{POV} := \{\texttt{driving}\}$

and $x \in \mathbb{R}$; and $B_1^{POV} := \Sigma^{POV*}$ for set of event labels $\Sigma^{POV} = \{\beta_1, \beta_2\}$. The model semantics are $[\![P_0]\!]^{B_0}$: the set of all hybrid traces with the discrete location `driving` and x that starts in the initial condition set $[-420, -300]$ and evolves along any arbitrary derivative in the range $[\underline{v}_x, \overline{v}_x]$, and $[\![P_1]\!]^{B_1}$: the singleton set $\{\beta_1\beta_2\}$.

A behavior abstraction function $\mathcal{A}^{POV} : B_0^{POV} \to B_1^{POV}$ constructed by partitioning the continuous dimension x at boundaries $x = l$ and $x = 0$ is written mathematically as follows. Given $b_0^{POV} = h^{POV}(t) \in B_0^{POV}$ and $b_1^{POV} = \sigma_0\sigma_1 \cdots \in B_1^{POV}$, $\mathcal{A}^{POV}(b_0^{POV}) = b_1^{POV}$ iff $\exists$ times $t_i \in \mathbb{R}_+$ s.t. $\forall t' \in [0, t_0)$, $x(t') \in \text{FROM}(\sigma_0)$; $\forall t' \in [t_{i-1}, t_i)$, $x(t') \in \text{TO}(\sigma_{i-1}) \cap \text{FROM}(\sigma_i)$ for $i = 1, \ldots, N$ for some $N \in \mathbb{N}$; and $\forall t' \geq t_N$, $x(t') \in \text{TO}(\sigma_N)$, where $\text{FROM}(\cdot)$ and $\text{TO}(\cdot)$ are given in the following table.

σ	$\text{FROM}(\sigma)$	$\text{TO}(\sigma)$
β_1	$x \leq l$	$x \in [l, 0]$
β_2	$x \in [l, 0]$	$x \geq 0$

Otherwise, $\mathcal{A}^{POV}(b_0^{POV}) = \varepsilon$.

Suppose the boundary l is at -300. Since the range of velocities is positive, and initial condition is in the range $[-420, -300]$, it is straightforward to show that $\forall b_0^{POV} \in B_0^{POV}$, $\mathcal{A}^{POV}(b^{POV}) = \beta_1\beta_2$. Therefore, $P_0 \sqsubseteq^{\mathcal{A}^{POV}} P_1$. Note that if l is say -310, $\mathcal{A}^{POV}(b^{POV}) = \beta_2$ for some b^{POV} and $P_0 \not\sqsubseteq^{\mathcal{A}^{POV}} P_1$.

6.5.2 Heterogeneous Abstraction for SV

The local behavior domains for the SV models Q_0 and Q_1 are B_0^{SV}: the set of 3-d hybrid trajectories $h^{SV}(t)$, where $h^{SV} := (l^{SV}, x, y_s, v_s)$ are the hybrid states that take values in $\mathcal{L}^{SV} \times \mathcal{X}^{SV}$, for the discrete set of locations $\mathcal{L}^{SV} := \{\texttt{waiting}, \texttt{stopped}, \texttt{conflict\_s}, \texttt{clear\_s}\}$ and the continuous state space $\mathcal{X}^{SV} := \mathbb{R}^3$; and $B_1^{SV} := \Sigma^{SV*}$ with $\Sigma^{SV} := \{\alpha_1, \alpha_2, \beta_1\}$, where α's signify SV entering and exiting the intersection.

A behavior abstraction function $\mathcal{A}^{SV} : B_0^{SV} \to B_1^{SV}$, constructed by only keeping the discrete part of the hybrid model and adding transition labels, is written formally as follows. Given $b_0^{SV} = h^{SV}(t)$, where $t \in \mathbb{R}_+$ and $h^{SV} = (l^{SV}, x, y_s, v_s)$, and $b_1^{POV} = \sigma_0\sigma_1\cdots$ with states $q_i^{SV} \in \mathcal{L}_i^{SV}$ s.t. $q_i^{SV} \xrightarrow{\sigma_i} q_{i+1}^{SV}$, $\mathcal{A}^{SV}(b_0^{SV}) = b_1^{SV}$ iff $\exists$ times $t_i \in \mathbb{R}_+$ s.t. $\forall\ t' \in [t_i, t_{i+1})$ with $t_0 = 0$, $l^{SV}(t') == q_i^{SV}$. Otherwise, $\mathcal{A}^{POV}(b_0^{POV}) = \varepsilon$.

Because Q_1 has the exact same discrete transition graph as that of Q_0, for every hybrid behavior $b_0^{SV} \in [\![Q_0]\!]^{B_0^{SV}}$, $\mathcal{A}^{SV}(b_0^{SV}) \in [\![Q_1]\!]^{B_1^{SV}}$, i.e., $Q_0 \sqsubseteq^{\mathcal{A}^{SV}} Q_1$.

6.5.3 Abstraction Between Compositions

At the discrete level of abstraction, the global unified behavior domain B_1 is Σ^*, where $\Sigma = \Sigma^{POV} \cup \Sigma^{SV} = \{\alpha_1, \alpha_2, \beta_1, \beta_2\}$. Behavior localizations $\downarrow_1^j$, $j = P, Q$ are discrete event projection functions that replace a string not in the local label set by the empty string ε. In this case, the syntactic procedures of adding self loops on the missing labels α_1, α_2 in P_1 and β_2 in Q_1 take care of the globalizations and their composition is simply their product. At the hybrid level, we add an unrestricted continuous variable y in P_0 leaving Q_0 unchanged, and take the parallel composition of the resulting hybrid automata.

The variables common to local behavior domains B_i^{POV} and B_i^{SV} are x and β_1. We have to make sure that the localizations $\mathcal{A}^{POV}\Downarrow^{\cap}$ and $\mathcal{A}^{SV}\Downarrow^{\cap}$ of abstraction functions $\mathcal{A}^{POV}$ and $\mathcal{A}^{SV}$ onto these common variables , i.e., the mappings from behaviors in x to behaviors in $\{\beta_1\}^*$ agree. $\mathcal{A}^{POV}\Downarrow^{\cap}$ is essentially the same as $\mathcal{A}^{POV}$, with the row for β_2 discarded. $\mathcal{A}^{SV}$ puts indirect restrictions on x due to the guard and invariant conditions of the hybrid transitions $(\texttt{waiting}, x) \to (\texttt{stopped}, x)$ that are mapped with the discrete transition $\texttt{waiting} \xrightarrow{\beta_1} \texttt{stopped}$. Such a hybrid transition occurs iff $x \leq l$ and $x \geq l$ hold before and after the transition, i.e., while crossing the boundary $x = l$ in the increasing direction, which agrees with $\mathcal{A}^{POV}\Downarrow^{\cap}$. In the self-loop β_1 transitions, x does not appear and is therefore unrestricted, and in agreement with $\mathcal{A}^{POV}\Downarrow^{\cap}$. Therefore, using compositional heterogeneous abstraction from Prop. 4.3, we can conclude $M_{41} \sqsubseteq^{\mathcal{A}} M_{53}$.

In summary, we have used compositional heterogeneous abstraction from Sec. 4.2 to use abstraction between heterogeneous component models using local behavior abstraction functions to conclude abstraction between the composite models. The consistency condition for distributed heterogeneous abstraction works out. Note that if for some reason, the parameter l is different in models P_0 and Q_0, the consistency condition in Prop. 4.3 cannot be satisfied and the heterogeneous approach cannot be used. Suppose the reference marker in the POV component is l' rather than l, but the SV thinks it is l. Physically this may correspond to, e.g., a measurement error or parallax for a human SV driver. In this case, there is a disagreement between the two models as to what corresponds to the β_1 event of POV going from `far` to `close`. Since in this case the two abstraction functions would disagree on the mapping between behaviors in the variable x and the event β_1 that

are common to the local behavior domains of the two components, the design freedom in the non-uniqueness of globalizations while adding the remaining variables and events would not help us resolve this mismatch. These kinds of mismatches can be avoided by enforcing the consistency conditions for the behavior abstraction functions for decentralized development from Sec. 4.2.

6.6 Consistent Parametric Heterogeneous Verification

In this section, we illustrate the use of consistent parametric verification construct developed in Sec. 5.4. Consistent parametric verification uses the parametric extensions of semantic interpretations, abstraction and entailment, and uses external-constraint consistency to guarantee consistent heterogeneous verification.

To illustrate the use of parametric heterogeneous verification, we return to the conjunctive heterogeneous verification of model M_{41} by models M_{5i}. The parameterized models are shown in Fig. 6.11 and 6.12. The parameterized specifications are

- $S_{41} : \Box\neg(x == 0 \wedge 0 \leq y_s \leq h)$
- $S_{51} : \Box(x == l \Rightarrow \Box_{t_x}\ x < 0)$
- $S_{52} : \Box(\Diamond_{t_y} y_s \geq h)$
- S_{53} stays the same as the unparameterized one.

The bounds on the POV velocity, the bounds on the SV acceleration, the position of the marker l and the lane width of the major road h are represented as parameters as shown in Fig. 6.11 and Fig. 6.12. These parameters embedded in the unparameterized

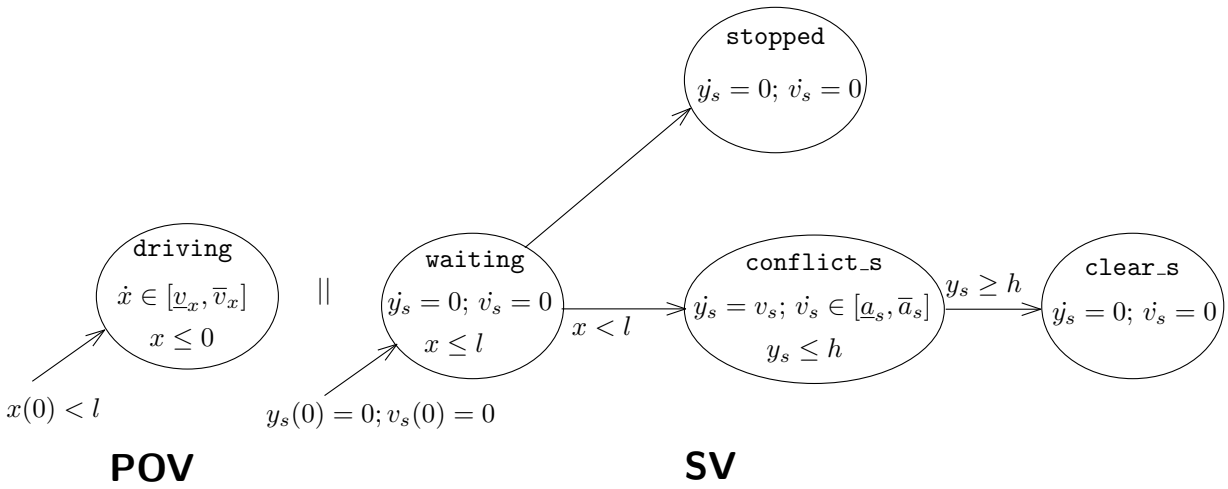
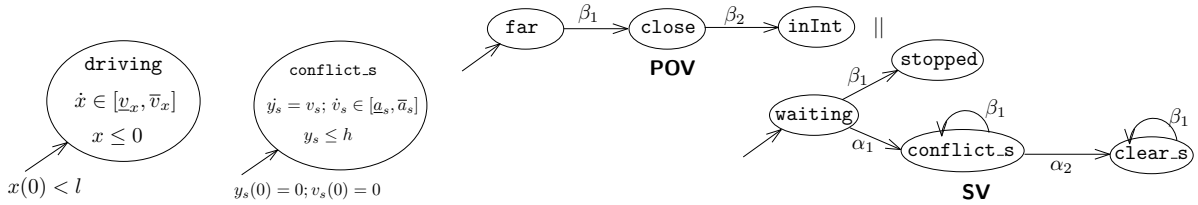


Figure 6.11: A parameterized hybrid model M_{41} for SV going only straight if safe.



(a) Parameterized model M_{51} (b) Parameterized model M_{52} (c) Model M_{53}

Figure 6.12: Parameterized models M_{5i}.

models are now explicitly identified as follows.

- $P_{41}^M : \{M_{41}.\underline{v}_x, M_{41}.\overline{v}_x, M_{41}.l, M_{41}.h, M_{41}.\underline{a}_y, M_{41}.\overline{a}_y\}$,
- $P_{41}^S : \{S_{41}.h\}$,
- $P_{51}^M : \{M_{51}.\underline{v}_x, M_{51}.\overline{v}_x, M_{51}.l\}$,
- $P_{51}^S : \{S_{51}.l, S_{51}.t_x\}$,
- $P_{52}^M : \{M_{52}.h, M_{52}.\underline{a}_y, M_{52}.\overline{a}_y\}$,
- $P_{52}^S : \{S_{52}.h, S_{52}.t_y\}$,
- $P_{53}^M : \{\}$,

- $P^S_{53} : \{\}$.

The following constraints identify the ranges of these parameters.

- $C^M_{41} : 20 \leq M_{41}.\underline{v}_x \leq M_{41}.\overline{v}_x \leq 30 \wedge M_{41}.l == -300 \wedge M_{41}.h == 4.5 \wedge 0.25 \leq M_{41}.\underline{a}_y \leq M_{41}.\overline{a}_y \leq 5$
- $C^S_{41} : S_{41}.h == 4.5$
- $C^M_{51} : 18 \leq M_{51}.\underline{v}_x \leq M_{51}.\overline{v}_x \leq 32 \wedge M_{51}.l == -300$
- $C^S_{51} : S_{51}.l == -300 \wedge 9 \leq S_{51}.t_x \leq 10$
- $C^M_{52} : M_{52}.h == 4.5 \wedge 0.2 \leq M_{52}.\underline{a}_y \leq M_{52}.\overline{a}_y \leq 5.2$
- $C^S_{52} : S_{52}.h == 4.5 \wedge 7 \leq S_{52}.t_y \leq 8$

Now, we know that the time needed for the POV to get from l to 0 needs to be bigger than the time needed for the SV to start accelerating from a stationary position and clear the intersection (i.e., $t_y < t_x$). From Newton's laws of motion, we note that $\sqrt{\frac{2h}{\underline{a}_y}} \leq t_y$ and $t_x \leq \frac{-l}{\overline{v}_x}$. We add this to C_{aux} along with the equality constraints between the parameters that are identical between M_{5i}s and M_{41}:

$$C_{\mathsf{aux}} : \quad (M_{41}.\underline{v}_x == M_{51}.\underline{v}_x) \wedge \ldots \wedge (M_{41}.\overline{a}_y == M_{52}.\overline{a}_y) \wedge$$
$$(\sqrt{\frac{2h}{\underline{a}_y}} \leq t_y < t_x \leq \frac{-l}{\overline{v}_x})$$

We have a parametric abstraction for each model because due to the equality constraints in C_{aux}, we get equal parameter valuations for the corresponding models, and under the same parameter valuations, M_{5i} are not more restrictive than M_{41}. Note that we have parametric conjunctive specification implication so long as $t_y < t_x$ holds, and here it does.

$C^M_{51}, M_{51} \models^{B_{51}} C^S_{51}, S_{51}$ and $C^M_{52}, M_{52} \models^{B_{52}} C^M_{52}, S_{52}$ can be shown using Newton's laws so long as $\sqrt{\frac{2h}{\underline{a}_y}} \leq t_y$ and $t_x \leq \frac{-l}{\overline{v}_x}$ hold, which they do. $C^M_{53}, M_{53} \models^{B_{53}} C^M_{53}, S_{53}$ still holds since it hasn't changed from the unparameterized case.

Finally, we get the following projections of C^M_{41} and C^S_{41} on P^M_{5i} and P^S_{5i} through C_{aux}:

- $(C^M_{41} \wedge C_{\mathsf{aux}}) \downarrow_{P^M_{51}}$: $20 \leq M_{51}.\underline{v}_x \leq M_{51}.\overline{v}_x \leq 30 \wedge M_{51}.l == -300 \wedge M_{51}.\overline{v}_x < 33.33$
- $(C^S_{41} \wedge C_{\mathsf{aux}}) \downarrow_{P^S_{51}}$: $\top$
- $(C^M_{41} \wedge C_{\mathsf{aux}}) \downarrow_{P^M_{52}}$: $0.25 \leq M_{52}.\underline{a}_y \leq M_{52}.\overline{a}_y \leq 5 \wedge M_{52}.h == 4.5 \wedge M_{52}.\underline{a}_y > 0.19$
- $(C^S_{41} \wedge C_{\mathsf{aux}}) \downarrow_{P^S_{52}}$: $M_{52}.h == 4.5$

We have $(C^M_{41} \wedge C_{\mathsf{aux}}) \downarrow_{P^M_{51}} \Rightarrow C^M_{51}$, $C^S_{51} \Rightarrow (C^S_{41} \wedge C_{\mathsf{aux}}) \downarrow_{P^S_{51}}$; and $(C^M_{41} \wedge C_{\mathsf{aux}}) \downarrow_{P^M_{52}} \Rightarrow C^M_{52}$, $C^S_{52} \Rightarrow (C^S_{41} \wedge C_{\mathsf{aux}}) \downarrow_{P^S_{52}}$. We have also proved these semantic consistency conditions in the theorem prover KeYmaera [75]. Now we can use parametric conjunctive heterogeneous verification and conclude that $C^M_{41}, M_{41} \models^{B_{41}} C^S_{41}, S_{41}$.

In this parameterized example, because we are able to capture the parameter dependencies, we now know how fast the SV needs to accelerate given ranges of $\overline{v}_x$, h and l. Alternatively, if the system is implemented as a road-side infrastructure-based solution, where $\underline{a}_y$ cannot be chosen but is known empirically from driver behavior data, we know how l should be chosen. While the heterogeneous verification of the unparameterized example succeeds, there is no support for capturing these interdependencies. Therefore, there is value added in exposing parameters and identifying interdependencies.

Although we have illustrated the use of parametric heterogeneous verification for only one node, verification constructs at other nodes can be performed using explicitly identified parameters and auxiliary constraints for interdependencies from all across the tree in a

similar manner. For example, the communication delay τ_c, the driver response time τ_d and the computation time τ_{cc} introduced in Sec. 6.4.1 can be added to the auxiliary constraint by modifying it as

$$(\sqrt{\frac{2h}{\underline{a}_y}} \leq t_y) \wedge (t_y + \tau_c + \tau_d + \tau_{cc} < t_x) \wedge (t_x \leq \frac{-l}{\overline{v}_x}).$$

This would ensure that the margin of difference between t_x, the worst case time-to-intersection for the POV, and t_y, the time-to-clear-intersection for the SV, overapproximates all the delays in the system. The sensor readings received are slightly delayed due to the communication links, the computation takes some time to finish, and the driver takes some time to respond. Given these several delays, the objective is to ensure that there is still enough time left for the SV to safely clear the intersection before the POV reaches the intersection. This new auxiliary constraint along with the knowledge about the ranges of these delay parameters would provide a new set of external constraints when projected onto the local sets of parameters for the parametric verification tasks of the SV and the POV models.

	Theoretical Concepts Illustrated	Theory Section
Sec. 6.2	Hierarchical verification tree	Sec. 5.5
Sec. 6.3	Disjunctive verification	Sec. 5.3.2
Sec. 6.3.1	Model coverage using behavior relations	Sec. 4.1
Sec. 6.3.2	Coverage with mode switching	Sec. 4.1
Sec. 6.4	Conjunctive verification	Sec. 5.3.1
Sec. 6.4.1	Heterogeneous abstraction using a common behavior domain	Sec. 4.1
Sec. 6.4.2	Hetetogeneous verification using behavior relations	Sec. 4.1
Sec. 6.5	Compositional heterogeneous abstraction	Sec. 4.2
Sec. 6.6	Consistent Parametric Verification	Sec. 5.4

Table 6.1: Illustration of theoretical concepts in the CICAS-SSA case study.

6.7 Summary

In this chapter, we have used the hierarchical heterogeneous verification of a cooperative intersection collision avoidance system as a case study to demonstrate the practical use of the theoretical concepts developed in this thesis. Table 6.1 summarizes the theoretical concepts illustrated in each section of this chapter and points out references to the corresponding places in the earlier chapters where the theory is developed.

We have illustrated that heterogeneity can be addressed by defining semantics in a common behavior domain as well as by associating heterogeneous domains using behavior relations. Heterogeneous abstractions via common semantic domains or behavior relations are used in conjunctive heterogeneous verification. Model coverage as well as coverage for mode switching is used in disjunctive heterogeneous verification. Compositional heterogeneous abstraction of hybrid and discrete component models is performed in a distributed manner. Semantic interdependencies are modeled using constraints over parameters and consistency is ensured using parametric extensions of heterogeneous verification.

The theoretical concepts illustrated at various places in the case study are used towards a system-level verification of the CICAS-SSA, represented at the root node, Node 01, in Fig. 6.3. The bounds on communication, driver response and computation delays, and measurement error bounds from the analyses at Nodes 11, 12, 13 and 15 are used indirectly within the heterogeneous verification of the hybrid-dynamic model at Node 13. Starting with the model at Node 13, at each subsequent level of abstraction, conjunctive or disjunctive analysis breakdowns or coverage for mode-switching are used to make the verification task iteratively simpler with each step. Rigorous definitions of behavior domains,

behavior relations (or abstraction functions), model abstractions or coverage, and individual or conjunctive specification implication are used to establish the sufficient conditions that let us infer verification at each node based on the simpler verification tasks at their children nodes. Ultimately, the verification performed on simple continuous and discrete models at the leaf nodes of the verification tree can be used to conclude more involved analysis tasks at their ancestors, involving higher dimensional hybrid dynamics. Bounds on various delays, intersection geometry, conflict zone boundaries for different lanes, and bounds on accelerations and velocities can be introduced as parameters and dependencies can be defined using auxiliary constraints. Semantic external-constraint consistency is established using analysis tools, such as the theorem prover KeYmaera.

Some methodological observations and experience from this case study can be useful for future heterogeneous verification activities, at least at a conceptual level. Starting with the system-level verification objective at the root node of a verification tree, the first step of conjunctively using models for different aspects of the system enables separation of concerns. Formal verification is known to be computationally expensive, therefore this early separation of other aspects from the verification model has value in keeping the analysis complexity at check. Given this conjunctive breakdown, the use of individual specifications to get bounds on parameters of interest such as delays, errors and physical limits, allows one to check only the properties of interest for that particular design aspect without having to worry about the other aspects or the system-level verification objective. The use of these bounds via auxiliary parameter constraint enables a mechanism to combine the analysis results from these aspects towards the system-level analysis. Hierarchical decompositions

seem to be the approach most useful for simplifying the verification objective in detailed verification model(s); however, it could be equally effective in reducing analysis complexity for other models in real-world complex systems. The use of different abstractions according to the particular cases (disjunctive analyses), modes (coverage with mode switching) and aspects (conjunctive analyses) can be made as per the specifics of the examples.

From an engineering workflow perspective, the separation of concerns in terms of models of different aspects of system design, such as physical dynamics, software, network, and abstract verification models, enables different analyses for these models to be performed by different groups or experts in parallel and in relative isolation from one another. Abstraction hierarchies for individual models can be constructed based on the needs and the specifics of only the individual problem at hand, as per the observations listed in the earlier paragraph, which can remain local to the group involved in developing that particular aspect of the system design. Formal definitions of behavior domains, and behavior relations for specific aspects allow the design experts of these different aspects to know the implications of their designs at the system level. Parameter constraints and external-constraint consistency conditions can be used at the time of system integration by facilitating the composition of the analysis results from these independently-developed aspects in a semantically sound and consistent manner.

Chapter 7

Conclusion

This chapter summarizes the contributions of the thesis and outline some future research directions.

7.1 Summary of the Contributions

The contributions of this thesis are as follows.

1. *Behavioral semantics for heterogeneous models and specifications.* To address the problem of heterogeneity in model-based design of CPS, we have created a general framework based on behavioral semantics. The generality of our framework permits the use of several modeling and specification formalisms, analysis techniques and tools because no assumptions are made about the specifics of any particular formalism.

   The use of behavioral semantics also lets us define the semantic interpretations of models and specifications in several different behavior domains. This provides a basis

for comparing the semantic interpretations of models and specifications as subsets of a common behavior domain.

The use of behavioral semantics offers a uniform semantic treatment to models and specifications in terms of the sets of behaviors in a given domain that they exhibit or allow. This identical treatment allows us to develop similar notions of abstraction, entailment and implication, all as behavior set inclusions of corresponding semantic interpretations of the models and specifications.

2. *Semantic associations between heterogeneous formalisms using behavior relations.* To support heterogeneity in the semantic definitions of models and specifications from suitable behavior formalisms, we allow semantics to be defined across different behavior domains. This is particularly useful when some analysis techniques and tools leverage particular behavior formalisms best. When semantic interpretations of models and specifications are defined in such heterogeneous behavior domains, behavior relations provide the associations between these domains. This gives us the ability to compare subsets of behaviors that are the semantic interpretations of various models and specifications in heterogeneous behavior domains.

3. *Abstractions across heterogeneous modeling and semantic formalisms using behavior relations.* Abstractions are necessary to make formal analysis tractable. In case of CPS, abstractions across heterogeneous modeling formalisms are necessary. Abstraction across different modeling formalisms can be defined using semantic interpretations in a common behavior domain when a model overapproximates the set of possible behaviors of another model. When semantics of models are defined in

different heterogeneous behavior domains, the definition of abstraction can be extended to include associations using behavior relations to relate sets of behaviors to be overapproximated.

Often times it is necessary to use multiple different abstractions in different modeling formalisms together to analyze different aspects of system design. When different operating regimes of systems are best captured using different models, a notion of disjunctive abstraction, or *coverage*, is developed to ensure that each behavior of the system is considered in at least one model. For mode switching systems that are analyzed for temporally invariant specifications, a notion of coverage with switching is also developed with correct overapproximation of initial conditions such that individual modes can be analyzed independently.

4. *Compositional approach to heterogeneous abstraction using behavior abstraction functions as special cases of behavior relations.* Complex systems are often composed of subsystems, and it is beneficial to analyze subsystems independently for supporting separation of concerns. We have developed a framework for formally defining semantic composition of component models using behavioral semantics. In general, the behavior domains that define semantics of subsystem models can be different from each other, where only the local semantics of component models are considered. A notion of globalized semantic composition is developed to lift the local semantics of subsystem models to a common domain where they can be composed.

   For analyzing heterogeneous abstractions compositionally, arbitrary behavior relations turn out to be insufficient to guarantee compositionality. Instead, special

cases of behavior relations called *behavior abstraction functions* are used. Centralized development can be used to break down a system-level abstraction problem into subsystems. On the other hand, distributed development can be used to construct suitable consistent abstraction functions such that abstraction established independently for subsystems also holds for the composition.

5. *Implications between heterogeneous specifications and their use for verification using behavior relations.* Specifications are useful for defining correctness requirements for systems and their models. This thesis considers safety specifications and defines the set of allowed behaviors in a given behavior domain as the semantic interpretation of such specifications. Analogous to abstraction, implication between heterogeneous specifications can be supported by defining semantics either in a common domain or across heterogeneous behavior domains via behavior relations. Multiple specifications defined for multiple models can be used together in a conjunctive implication of the system-level specification to guarantee that what gets specified as requirements for individual models implies the requirement for the underlying system.

   Entailment, defined as the set of behaviors of a model being contained inside the set of behaviors allowed by a specification, can be established using several types of analysis tools depending on the context, such as state-space exploration approaches like reachability analysis, theorem proving, model checking, establishing certificate-based guarantees and exhaustive simulation.

6. *Hierarchical heterogeneous verification via behavior relations using nested conjunctive and disjunctive analysis constructs.* When several different models in different

formalisms get built and analyzed against different specifications, we have developed two constructs for putting together the individual analysis results in a meaningful way. Conjunctive heterogeneous abstraction uses individual abstraction by models and conjunctive specification implication, and disjunctive heterogeneous verification uses disjunctive model abstraction (coverage) along with individual specification implication. Disjunctive heterogeneous verification can also be used for mode-switching systems when specifications are not temporally invariant. Conjunctive and disjunctive heterogeneous verification constructs can be combined hierarchically in a tree structure to put together complex analysis hierarchies.

7. *Consistent heterogeneous verification in presence of semantic interdependencies.* In order to formally represent dependencies that exist across models from different formalisms, we have developed the use of model and specification parameters as a formal mechanism to capture the interdependencies in the assumptions made in the different modeling and specification formalisms. The use of auxiliary constraints to define interdependencies enable us to develop notions of semantic consistency for verification by projecting the dependencies and the system-level valuations of parameters onto the model and specification parameters of every analysis task at hand. This ensures that despite the interdependencies, what is being checked for the individual models still guarantees what is being inferred about the system.

8. *Demonstration of practical applicability of the approach using a case study.* We have demonstrated the applicability of the theoretical framework developed in this thesis for multi-model hierarchical heterogeneous verification of the cooperative intersec-

tion collision avoidance system for stop-sign assist (CICAS-SSA). CICAS-SSA is typical of the kinds of systems targeted in this thesis—it is heterogeneous, due to the constituent sensing, communication, computation and physical dynamics; and it is safety critical, which warrants formal verification of the system using the several heterogeneous models developed for the system. We have demonstrated a hierarchical heterogeneous verification tree that uses the individual pieces developed during the thesis—conjunctive and disjunctive heterogeneous verification, coverage for mode switching, compositional heterogeneous abstraction and parameter consistency.

7.2 Directions for Future Work

The following paragraphs outline some future research directions that were identified as logical next steps during the development of the theoretical framework and the case study.

1. *Creation of state-based relations for constructing behavior relations.* The framework developed in this thesis requires relations between behavior domains for supporting heterogeneity. The specific behavior relations used for illustrations in this thesis are inspired from some commonly used abstraction techniques such as state-space partitioning, projecting away continuous variables, considering only subparts of trajectories in specific modes, and ensuring the same discrete and hybrid state transition graphs. While associations are necessary at the behavior level, often times conditions or constraints on states are developed in order to guarantee operations for the behaviors, such as simulation relations on states to guarantee language inclusion.

While these methods are supported in our framework, we have not developed general mechanisms for defining state-based mappings that would lead to the construction of behavior-based mappings. Heterogeneous generalizations of simulation relations [38], timed simulation relations [39] and other state-based mappings such as approximate (bi)simulation functions [44] and approximate syncrhonization [53] could be some specific starting points from which one can generalize. Developing such extensions would facilitate the use of our approach when it is easier to think at the level of states rather than entire behaviors.

2. *Heterogeneity at a given level of abstraction between interacting component models.* The compositional heterogeneous abstraction framework developed in this thesis only considers heterogeneity at two different levels of abstraction. Within a given level of abstraction, heterogeneity between component models modeled in different formalisms can currently be addressed by defining their semantics in a common behavior domain. A possible extension to this work is to permit heterogeneity between the respective semantic domains of the component models at a given level of abstraction. Heterogeneous generalizations of composition and localization/globalization are needed in order to define *heterogeneous globalized semantic composition*.

   A starting point could be the framework of Benveniste et al. for composing heterogeneous reactive systems [19] in which heterogeneous parallel composition is defined using an algebra of tag structures following the tagged-signal semantics of Lee and Sangiovanni-Vincentelli [62]. A similar notion of heterogeneous parallel composition could be developed within the behavioral semantics framework developed in this

thesis.

3. *Translation between models and specifications.* Our identical semantic treatment of models and specifications in terms of their corresponding sets of behaviors leads to some interesting possibilities that could be of value in some applications. We note that abstraction, entailment and implication are all mathematically inclusions of behavior sets possibly via behavior relation mappings. These semantically equivalent notions are syntactically very different given the difference between the operational (direct) and declarative (indirect) descriptions typically used for models and specifications, respectively. The tool support for these different flavors of the same semantic problem also varies a great deal. Tools such as PHAVer [40] are good at establishing abstraction between two models, tools such as CheckMate [29] are good at establishing entailment between a model and a specification, while theorem provers such as KeYmaera [75] are good at proving implications between specifications. Translation between models and specifications would enable system designers to leverage the power of exploiting syntactically easier formalisms to deal with the semantically equivalent problems. The translation from a specification to a model, which we call *modelization*, is already used in tools such as SPIN [50] and LTSA [64] because it is easy to translate temporal logic specifications into equivalent transition system models. For various pairs of heterogeneous modeling and specification formalisms, it would be worth investigating whether such translation leads to more efficient algorithms for establishing heterogeneous behavior set inclusion for abstraction, implication and entailment.

4. *Semantic consistency using variable constraints.* This thesis uses constraints over static parameters as a mechanism to formally represent semantic interdependencies. A natural extension of constraints on parameters is to constraints on variables that can change over time. However, we lose the generality across modeling and specification formalisms provided by the parameter constraints due to the unambiguous and uniform semantic interpretations of models and specifications that use these parameters. Constraints over variables can be expressed in temporal or dynamic logics and the notions of projection using existential quantification and implication still can be used. However, semantic interpretations of models and specifications in presence of certain assumptions about dynamically changing variables need to be defined.

   A starting point could be to use modelization to turn the variable constraints (which can be thought of as specifications) into abstract models to define composition. In this case, the consistent parametric verification problem has the form of heterogeneous assume-guarantee reasoning, where the constraints over variables are used as assumptions about the environment for guaranteeing properties. Further investigation and evaluation on some examples with time-varying interdependencies would be useful.

5. *Integrating the multi-view architectural framework and the multi-model heterogeneous verification framework.* Bhave et al. use architectural modeling of cyber-physical systems as a unifying high-level structural representation of systems. Graph morphisms are used to ensure structural consistency between architectures of heterogeneous models, called *views*, and the underlying base architecture of the system. We

have proposed the use of a multi-view architectural framework for managing parameter consistency [78]. Recently we have also been investigating a combined approach of using the multi-view architectural approach and multi-model heterogeneous verification approach together towards system design. Our preliminary work in this direction appears in work submitted for publication [79].

The structural information contained in architectures can be used to exploit richer substructure decomposition for compositional heterogeneous abstraction part of our framework. The knowledge from the structural information in the architectures can also lead to simplifying assumptions or decompositions that make verification obligation easier. A specific example of using this type of structural information from the case study is the observation that the oncoming vehicles arrive at the intersection in order, and once a vehicle crosses the intersection, the next one becomes the new POV. This observation enables the coverage of multi-vehicle major road models by single-POV models. Principled approaches to capture and exploit such connections between the structural and semantic sides need to be developed and explored further.

6. *Temporally varying specifications for mode-switching disjunctive coverage.* In this thesis we have introduced the concept of coverage for using different models for different modes of mode-switching systems. However, this approach works only for specifications that are temporally invariant. For more general temporally varying specifications, we could extend our work to include notions similar to that of the so-called verification architectures introduced by Faber [37]. Verification architectures use high-level switching protocols to verify an overall system property, and

require that each mode adheres to this high-level protocol. A similar approach can be explored in the general setting of behavioral semantics.

7. *Creation of abstractions from detailed simulation models.* Detailed simulation models in commercial tools such as Simulink are often created in industry to simulate complex cyber-physical systems and generate embedded software code. The first hurdle in analyzing these existing models using verification capabilities of different tools is usually to construct correct heterogeneous abstractions of the simulation models using the formalisms supported by these tools. Recently, an approach to reverse engineer requirement specifications from simulation models, called *specification mining*, has been developed by researchers at Toyota [51]. Specification mining constructs tight overapproximations of simulation models. These mined specifications can be used with modelization approaches to construct models in formalisms of interest. This combination of mining and modelization could provide a principled approach to constructing heterogeneous abstractions from an existing simulation models.

8. *Bridging the gap between theoretical developments and their practical applicability.* This thesis has made theoretical contributions towards addressing the heterogeneity, consistent resolution of interdependencies, abstraction across different formalisms and support for compositional reasoning in centralized and decentralized development of subsystems in a general and formal framework. All these research challenges are motivated from the challenges in practice, arising in model-based development of cyber-physical systems. However, the adoption of the theoretical machinery from this thesis by practicing engineers would require understanding of how their under-

standing of models, what they mean, and their relations and interdependencies would be formally defined in our framework of behavior semantics, behavior relations. Insight about where and when to use various types of breakdowns such as conjunctive, disjunctive and mode-switching coverage, and local and global semantics and their use for compositional development would also need to be provided. The CICAS-SSA case study presented in this thesis aims to demonstrate the practical applicability of the theoretical concepts. Some methodological observations from that exercise outlined in the summary of the case study chapter can be generalized as a guidance for use in similar exercises in other domains, as outlined next.

A potential starting point towards the adoption of this work by practicing engineers would be to define a library of definitions of behavior domains in various formalisms. In this thesis, we have used domains defined in continuous trajectories, piecewise-continuous trajectories, discrete traces and hybrid trajectories of various dimensions. Such a library of formal definitions of behavior domains could be useful for practicing engineers for formally defining the semantics of their models.

This thesis uses specific behavior relations inspired from some commonly used abstraction techniques such as state-space partitioning, projecting away continuous variables, considering only sub-parts of trajectories in specific modes, and ensuring the same discrete and hybrid state transition graphs, and a potential future direction of the use of state-based relations for constructing behavior relations has been mentioned earlier in this list. A library of such commonly used behavior relations would be useful in formally defining how these types of mappings are implicitly constructed

in the reasoning in practice.

Finally, while the specific choice of breakdowns used for abstraction hierarchies would always be problem-specific, the insight gained from the CICAS-SSA case study about what breakdowns are useful in what contexts can be made available as a potential source for trying out similar breakdowns in other contexts. Conjunctive abstraction is a common choice while representing different aspects of the system design such as computation, communication and physical dynamics. Disjunctive breakdown would be most useful when there are distinct subsets of system behaviors, such as different cases of a non-deterministic choice, or linear vs. nonlinear dynamics, etc., where different cases can be better analyzed using different abstractions. Coverage for mode-switching systems is useful when different modes have distinctly different dynamics. For example, in case of powertrain models, idle-speed mode has a closed-loop control of idle speed around a set point; during nominal operation, the open-loop choice of throttle control is made according to the stoichiometric ratio; while the full-load driving and braking modes have fully open and fully closed throttles respectively. These are distinctly different modes of operations that can be better analyzed in isolation using mode-switching coverage for temporally invariant specifications. A future direction towards supporting temporally varying specifications in mode-switching coverage has been mentioned earlier in this list. These guidelines can be useful at a conceptual level while deciding what breakdown to use in different contexts of system analyses.

Bibliography

[1] Cooperative intersection collision avoidance systems (CICAS). `http://www.its.dot.gov/cicas/`.

[2] Cooperative intersection collision avoidance systems-stop sign assist (cicas-ssa). `http://www.dot.state.mn.us/guidestar/2006_2010/cicas.html`.

[3] MapleSim. `http://www.maplesoft.com/products/maplesim/`.

[4] Modelica. `http://www.modelica.org/`.

[5] OMNeT++. `http://www.omnetpp.org`.

[6] Simulink. `www.mathworks.com/products/simulink/`.

[7] B. Aldrich, A. Fehnker, P. H. Feiler, Z. Han, B. H. Krogh, E. Lim, and S. Sivashankar. Managing verification activities using SVM. In J. Davies, W. Schulte, and M. Barnett, editors, *Formal Methods and Software Engineering*, volume 3308 of *Lecture Notes in Computer Science*, pages 61–75. Springer Berlin / Heidelberg, 2004.

[8] M. Althoff, A. Rajhans, B. H. Krogh, S. Yaldiz, X. Li, and L. Pileggi. Formal verification of phase-locked loops using reachability analysis and continuization. In *Proceedings of the IEEE/ACM 2011 International Conference on Computer-Aided Design (ICCAD)*, San Jose, Nov 2011.

[9] R. Alur. Formal verification of hybrid systems. In *Proceedings of the 11th International Conference on Embedded Software (EMSOFT)*, 2011.

[10] R. Alur, T. Dang, and F. Ivancic. Predicate abstraction for reachability analysis of hybrid systems. *ACM Transactions on Embedded Computing Systems*, 5(1):152–199, 2006.

[11] R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43:139–152, 1996.

[12] R. Alur and T. A. Henzinger. Reactive modules. *Formal Methods in System Design 15:7-48, 1999*, 15:7–48, 1999.

[13] R. Alur, T. A. Henzinger, G. Laffarriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88:971–984, 2000.

[14] R. Alur, A. Kanade, S. Ramesh, and K. C. Shashidhar. Symbolic analysis for improving simulation coverage of simulink/stateflow models. In *Proceedings of the 8th ACM & IEEE International Conference on Embedded Software (EMSOFT)*, 2008.

[15] E. Asarin, P. Caspi, and O. Maler. Timed regular expressions. *J. ACM*, 49(2):172–206, Mar. 2002.

[16] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. Sangiovanni-Vincentelli. Metropolis: an integrated electronic system design environment. *Computer*, 36(4):45–52, april 2003.

[17] A. Basu, S. Bensalem, M. Bozga, J. Combaz, M. Jaber, T.-H. Nguyen, and J. Sifakis. Rigorous component-based system design using the BIP framework. *IEEE Software*, 28(3):41–48, 2011.

[18] S. Bensalem, M. Bozga, T.-H. Nguyen, and J. Sifakis. Compositional verification for component-based systems and application. *IET Software*, 4(3):181–193, 2010.

[19] A. Benveniste, L. P. Carloni, P. Caspi, and A. L. Sangiovanni-Vincentelli. Composing heterogeneous reactive systems. *ACM Transactions on Embedded Computing Systems*, 7(4), July 2008.

[20] S. S. Bhattacharyya, E. Cheong, and I. Davis. PTOLEMY II heterogeneous concurrent modeling and design in Java. Technical report, University of California, Berkeley, 2003.

[21] A. Bhave. *Multi-view consistency in architectural views for cyber-physical systems*. PhD thesis, Carnegie Mellon University, 2011.

[22] A. Bhave, D. Garlan, B. H. Krogh, A. Rajhans, and B. Schmerl. Augmenting software architectures with physical components. In *Proc. of the Embedded Real Time Software and Systems Conf. (ERTS^2 2010)*, 19-21 May 2010.

[23] A. Bhave, B. H. Krogh, D. Garlan, and B. Schmerl. View consistency in architectures for cyber-physical systems. In *Proc. of Second International Conference on Cyber-Physical Systems,ICCPS*, 2011.

[24] S. Bliudze and J. Sifakis. The algebra of connectors - structuring interaction in BIP. *IEEE Trans. Computers*, 57(10):1315–1330, 2008.

[25] J. Chang, D. Cohen, L. Blincoe, R. Subramanian, and L. Lombardo. CICAS-V research on comprehensive costs of intersection crashes. In *Proceedings of the 20th International Technical Conference on the Enhanced Safety of Vehicles*, 2007. Paper Number 07-0016. Available at: `http://www-nrd.nhtsa.dot.gov/pdf/esv/esv20/07-0016-O.pdf`.

[26] K. Chaudhari, D. Doligez, L. Lamport, and S. Merz. The TLA+ proof system: building a heterogeneous verification platform. In *International Colloquium on Theoretical Aspects of Computing (ICTAC-7)*, volume LNCS 6256, page 44, Natal, Brazil, 2010.

[27] K. Chen. *A Semantic Anchoring Infrastructure for Model-Integrated Computing*. PhD thesis, Vanderbilt University, 2006.

[28] K. Chen, J. Sztipanovits, and S. Abdelwahed. Toward a semantic anchoring infrastructure for domain-specific modeling languages. In *5th ACM International Conference on Embedded Software*, volume September, 2005.

[29] A. Chutinan and B. H. Krogh. Verification of infinite-state dynamic systms using approximate quotient transition systems. *IEEE Transactions on Automatic Control*,

46:1401–1410, 2001.

[30] T. Dang, O. Maler, and R. Testylier. Accurate hybridization of nonlinear systems. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control (HSCC)*, 2010.

[31] J. Davis. GME: The Generic Modeling Environment. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, OOPSLA '03, New York, NY, USA, 2003. ACM.

[32] D. de Niz, G. Bhatia, and R. Rajkumar. Model-based development of embedded systems: the SysWeaver approach. *IEEE Real Time Technology and Applications Symposium*, pages 231–242, 2006.

[33] J. Dehlinger and R. Lutz. PLFaultCAT: A product-line software fault tree analysis tool. *Automated Software Engineering*, 13:169–193, 2006.

[34] T. Denton, E. Jones, S. Srinivasan, K. Owens, and R. W. Buskens. Naomi — an experimental platform for multi—modeling. In *Proceedings of the 11th international conference on Model Driven Engineering Languages and Systems*, MoDELS '08, pages 143–157, Berlin, Heidelberg, 2008. Springer-Verlag.

[35] A. Donzé, B. Krogh, and A. Rajhans. Parameter synthesis for hybrid systems with an application to simulink models. In *Proceedings of the ACM/IEEE Conference on Hybrid Systems: Computation and Control (HSCC)*, 2009.

[36] A. Donzé and O. Maler. Systematic simulation using sensitivity analysis. In *Proceedings of the 10th International Conference on Hybrid Systems: Computation and Control*, 2007.

[37] J. Faber. *Verification architectures for complex real-trime systems*. PhD thesis, University of Oldenburg, 2011.

[38] G. Frehse. *Compositional verification of hybrid systems using simulation relations*. PhD thesis, Radboud Universiteit Nijmegen, 2005.

[39] G. Frehse. On timed simulation relations for hybrid systems and compositionality. In *Proceedings of the 4th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2006.

[40] G. Frehse. PHAVer: Algorithmic Verification of Hybrid Systems past HyTech. *International Journal on Software Tools for Technology Transfer (STTT)*, 10(3), 2008.

[41] G. Frehse, C. L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: scalable verification of hybrid systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV)*, 2011.

[42] P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *Proceedings of the 13th International Conference on Computer Aided Verification*, CAV '01, pages 53–65, London, UK, UK, 2001. Springer-Verlag.

[43] R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proceedings of the Fifteenth IFIP WG6.1 International*

Symposium on Protocol Specification, Testing and Verification XV, pages 3–18, London, UK, UK, 1996. Chapman & Hall, Ltd.

[44] A. Girard, A. A. Julius, and G. J. Pappas. Approximate simulation relations for hybrid systems. *Discrete Event Dynamic Systems*, 18(2):163–179, 2008.

[45] A. Gorjestani, A. Menon, P.-M. Cheng, C. Shankwitz, and M. Donath. CICAS-SSA Report # 1: Alert and warning timing for CICAS-SSA. Technical report, Minnesota Guidestar, 2008.

[46] Z. Gu, S. Kodase, S. Wang, and K. Shin. A model-based approach to system-level dependency and real-time analysis of embedded software. In *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium, 2003.*, pages 78 – 85, May 2003.

[47] T. A. Henzinger. The theory of hybrid automata. *Verification of Digital and Hybrid Systems*, 170:292–296, 2000.

[48] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 43:225–238, 1998.

[49] T. A. Henzinger, S. Qadeer, S. K. Rajamani, and S. Tasiran. An assume-guarantee rule for checking simulation. *ACM Trans. Program. Lang. Syst.*, 24(1):51–64, Jan. 2002.

[50] G. J. Holzmann. *The SPIN model checker: primer and reference manual*. Addison Wesley, 2003.

[51] X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia. Mining requirements from closed-loop control models. In *Proceedings of the 16th ACM International Conference on Hybrid Systems: Computation and Control*, 2013.

[52] A. A. Julius. *On interconnection and equivalence of continuous and discrete systems: a behavioral perspective*. PhD thesis, University of Twente, 2005.

[53] A. A. Julius, A. D'Innocenzo, M. D. D. Benedetto, and G. J. Pappas. Approximate equivalence and synchronization of metric transition systems. *Systems & Control Letters*, 58(94-101):94–101, 2009.

[54] A. A. Julius, G. E. Fainekos, M. Anand, I. Lee, and G. J. Pappas. Robust test generation and coverage for hybrid systems. In *Proceedings of the 10th International Conference on Hybrid Systems: Computation and Control*, 2007.

[55] A. Kanade, R. Alur, F. Ivancic, S. Ramesh, S. Sankaranarayanan, and K. C. Shashidhar. Generating and analyzing symbolic traces of simulink/stateflow models. In *Proceedings of the 21st International Conference on Computer Aided Verification (CAV)*, 2009.

[56] D. Kaynar and N. Lynch. Decomposing verification of timed I/O automata. In *Proceedings of the Joint Conference on Formal Modelling and Analysis of Timed Systems (FORMATS) Formal Techniques in Real-Time and Fault Tolerant System (FTRTFT)*, 2004.

[57] Y. Kesten, Z. Manna, and A. Pnueli. Temporal verification of simulation and re-

finement. In J. de Bakker, W. de Roever, and G. Rozenberg, editors, *A Decade of Concurrency Reflections and Perspectives*, volume 803 of *Lecture Notes in Computer Science*, pages 273–346. Springer Berlin / Heidelberg, 1994.

[58] Y. Kesten and A. Pnueli. A compositional approach to verification. *Theoretical Computer Science*, 331:397–428, 2005.

[59] R. Kumar, B. H. Krogh, and P. Feiler. An ontology-based approach to heterogeneous verification of embedded control systems. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 370–385. Springer Berlin / Heidelberg, 2005.

[60] L. Lamport. What good is temporal logic? In *IFIP Congress*, pages 657–668, 1983.

[61] A. Ledeczi, J. Davis, S. Neema, and A. Agrawal. Modeling methodology for integrated simulation of embedded systems. *ACM Trans. Model. Comput. Simul.*, 13:82–103, January 2003.

[62] E. A. Lee and A. Sangiovanni-Vincentelli. A Framework for Comparing Models of Computations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(12):1217–1229, Dec 1998.

[63] J. Leung, T. Mandl, E. Lee, E. Latronico, C. Shelton, S. Tripakis, and B. Lickly. Scalable semantic annotation using lattice-based ontologies. In *12th International Conference on Model Driven Engineering Languages and Systems*, pages 393–407. ACM/IEEE, October 2009.

[64] J. Magee and J. Kramer. *Concurrency: State Models and Java Programming, Second Edition*. Wiley, 2006.

[65] M. Maile, F. Ahmed-Zaid, L. Caminiti, J. Lundberg, P. Mudalige, and C. Pall. Cooperative intersection collision avoidance system limited to stop sign and traffic signal violations: Midterm phase i report. Technical report, National Highway Traffic Safety Administration, 2008. Report No. DOT HS 811 048.

[66] O. Maler, D. Nickovic, and A. Pnueli. From MITL to timed automata. In E. Asarin and P. Bouyer, editors, *Formal Modeling and Analysis of Timed Systems*, volume 4202 of *Lecture Notes in Computer Science*, pages 274–289. Springer Berlin Heidelberg, 2006.

[67] Minesota Department of Transportation. CICAS-SSA: concept of operations.

[68] P. J. Mosterman and H. Vangheluwe. Computer automated multi-paradigm modeling in control system design. *IEEE Transactions on Control System Technology*, 12:65–70, 2000.

[69] T. Nghiem, S. Sankaranarayanan, G. E. Fainekos, F. Ivancic, A. Gupta, and G. J. Pappas. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In *Proceedings of the 13th International Conference on Hybrid Systems: Computation and Control*, 2010.

[70] G. Nicolescu and P. J. Mosterman. *Model-Based Design for Embedded Systems*. CRC Press, 2009.

[71] A. Pinto, L. P. Carloni, R. Passerone, and A. Sangiovanni-Vincentelli. Interchange

semantics for hybrid system models. In *Proceedings of the 5th International Conference on Mathematical Modeling*, 2006.

[72] A. Platzer. Using a program verification calculus for constructing specifications from implementations. Master's thesis, University of Karlsruhe, 2004.

[73] A. Platzer. Differential dynamic logic for hybrid systems. *J. Autom. Reas.*, 41(2):143–189, 2008.

[74] A. Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010.

[75] A. Platzer and J.-D. Quesel. KeYmaera: a hybrid theorem prover for hybrid systems. In A. Armando, P. Baumgartner, and G. Dowek, editors, *IJCAR*, volume 5195 of *LNCS*, pages 171–178. Springer, 2008.

[76] A. Pnueli. Specification and development of reactive systems (invited paper). In *IFIP Congress*, pages 845–858, 1986.

[77] J. Porter, P. Volgyesi, N. Kottenstette, H. Nine, G. Karsai, and J. Sztipanovits. An experimental model-based rapid prototyping environment for high-confidence embedded software. In *RSP '09: Proceedings of the 2009 IEEE/IFIP International Symposium on Rapid System Prototyping*, pages 3–10, Washington, DC, USA, 2009. IEEE Computer Society.

[78] A. Rajhans, A. Bhave, S. Loos, B. H. Krogh, A. Platzer, and D. Garlan. Using parameters in architectural views to support heterogeneous design and verification. In *50th IEEE Conference on Decision and Control*, Orlando, Dec 2011.

[79] A. Rajhans, A. Bhave, I. Ruchkin, B. H. Krogh, D. Garlan, and B. Schmerl. Supporting heterogeneity in cyber-physical system architectures. *Submitted to the IEEE Transactions in Automatic Control, Special Issue on Control of Cyber-Physical Systems*.

[80] A. Rajhans, S.-W. Cheng, B. Schmerl, D. Garlan, B. H. Krogh, C. Agbi, and A. Bhave. An architectural approach to the design and analysis of cyber-physical systems. In *Third International Workshop on Multi-Paradigm Modeling*, Denver, Oct 2009.

[81] A. Rajhans and B. H. Krogh. Heterogeneous verification of cyber-physical systems using behavior relations. In *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*, 2012.

[82] A. Rajhans and B. H. Krogh. Compositional heterogeneous abstraction. In *Proceedings of the 16th ACM International Conference on Hybrid Systems: Computation and Control*, 2013.

[83] S. Sankaranarayanan and G. E. Fainekos. Falsification of temporal properties of hybrid systems using the cross-entropy method. In *Proceedings of the 15th International Conference on Hybrid Systems: Computation and Control*, 2012.

[84] F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *Proceedings of the 12th International Conference on Computer Aided Verification*, CAV '00, pages 248–263, London, UK, UK, 2000. Springer-Verlag.

[85] K. Sullivan, J. Dugan, and D. Coppit. The Galileo fault tree analysis tool. In *Fault-Tolerant Computing, 1999. Digest of Papers. Twenty-Ninth Annual International Symposium on*, pages 232 –235, 1999.

[86] H. L. M. Vangheluwe. DEVS as a common denominator for multi-formalism hybrid systems modeling. In *Proceedings of the 2000 IEEE International Symposium on Computer-Aided Control System Design*, Anchorage, Alaska, USA, 2007.

[87] J. Willems. The behavioral approach to open and interconnected systems. *IEEE Control Systems Magazine*, 27(6):46–99, 2007.