MASTER

State space reduction for state-based and event-based models of Software Product Lines

Belder, T.

Award date:
2015

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Formal System Analysis Research Group

2IM91 - Master's Thesis

State space reduction for state-based and event-based models of Software Product Lines

Tessa Belder, BSc.

Supervisors:
dr. E.P. de Vink
dr.ir. T.A.C. Willemse

Final version

Eindhoven - August 31, 2015

Abstract

In times where there is an increasing demand for individualized software products, the popularity of Software Product Line Engineering (SPLE) as a software engineering paradigm is growing. As SPLE is being applied for the development of safety critical systems, being able to perform model checking for the verification of software product lines (SPL) is desirable.

This thesis considers both state-based and event-based models for software product families. For both of these models we define several equivalences, based on strong bisimulation (both worlds), branching bisimulation (event-based world), and divergence-blind stuttering equivalence (state-based world). Using these equivalences, we propose several minimal representations of the models.

Additionally, embeddings are proposed to convert between state-based and event-based models on the product-family level, and we show that at least one of the proposed equivalences is preserved and reflected by these embeddings. The embeddings allow to perform state space reduction on event-based models using state space reduction algorithms for state-based models and vice versa. This is beneficial in practice as it saves the costs of implementing and maintaining different algorithms for state space reduction in both worlds. Furthermore, these results indicate that both worlds may be equally expressive, as is commonly believed to be true on the single-product level.

Lastly, a toolset is developed to define SPL, to perform state space reduction on SPL, and to perform verification of the SPL using mCRL2. A small case study is performed on a vending machine toy example using the developed toolset. The results show that a product-family based verification approach in combination with product-family based state space reduction is much more effective than a regular enumerative method of verification, where each product is verified separately.

Preface

This master thesis is the result of my graduation project for the Computer Science and Engineering master program at Eindhoven University of Technology. The project was performed internally at the Formal System Analysis group of the Mathematics and Computer Science department of Eindhoven University of Technology.

First of all I would like to thank Erik de Vink for providing the opportunity to perform this project, for his help and guidance during the year that we have worked together, and for his everlasting patience. Furthermore I would like to thank him for the opportunity to write a paper together for the FMSPLE workshop, as well as for encouraging me to attend this workshop. I would like to thank Tim Willemse for guiding me during the second part of this project, and for his enthusiasm. I would like to thank both Erik and Tim for carefully reviewing my work many times, thereby providing valuable feedback. Furthermore my thanks go out to Maurice ter Beek for his contributions to our paper for the FMSPLE workshop.

I would also like to thank Petra, Hugo, Roel, Femke and Daniël, for their support and the many nice lunches during this project in particular, and for all the great times we had during our studies in general. Special thanks go out to Petra for kicking off my Japanese studies, which were a great distraction during the last part of this project.

And of course, my thanks go out to my mother and sister for always supporting me. I would like to thank Astrid in particular for always motivating me to perform better, making me exceed my own expectations.

Tessa Belder
July 2015

Contents

Chapter 1

Introduction

This master thesis is the final report of the graduation project for the Computer Science and Engineering master at Eindhoven University of Techonology (TU/e). The project is carried out within the Formal System Analysis group of the Mathematics and Computer Science department of TU/e.

This chapter introduces the research problem of this thesis. In Section 1.1 the context of this research problem is explained, and Section 1.2 describes the research problem itself. Finally, in Section 1.3 the obtained results are summarized, and an outline for the remainder of the thesis is provided.

1.1 Research problem context

Over the course of the past few decades software has become more prevalent in our daily lives. Not only have laptops, tablets, smart phones and personal computers become indispensable, but nowadays software is also present in devices that used to be purely mechanical, such as cars, household devices and vending machines. As people are becoming aware of the possibilities offered by this development, they are demanding their software products to become more and more individualized.

Companies developing software products try to fulfill this demand for individualization by building multiple slightly different versions of the same product. A common way to develop a new variation of an existing product is to copy and adapt this product. However, this way of working is rather incremental and often poorly documented, and hence companies end up with a set of products between which the connections are unclear. As a result it is likely that the same type of changes are implemented multiple times, possibly in a slightly different way each time, making this way of working expensive. Furthermore, it is both difficult and expensive to thoroughly test all the products. The latter is especially a problem for safety critical software, such as in cars, and for software that is produced in high volumes, such as in telephones. *Software product line engineering* [1, 25] (SPLE) is a method of software development proposed to solve these problems. In SPLE, a set of software products, referred to as the *software product line* (SPL) or *product family*, is developed simultaneously. By doing so, code for functions that are shared between (some of) the products can be reused, which results in lower production costs. Furthermore it allows for efficient implementation and testing.

For safety critical systems in particular, it is common practice to develop models of the behavior of a system before actually producing software. The process of checking whether such a model satisfies all requirements of the system is called *model checking*. An advantage of performing model checking is the ability to discover faults in the modeled behavior before implementing anything. Once a model is developed that satisfies all requirements it is possible to develop software based on this model. If this is done correctly it is likely that the developed software contains less bugs than it otherwise would have. As a drawback, model checking may be methodologically and computationally expensive, making it difficult to apply in practice for the verification of complicated software systems.

The behavior of a system is also called the *process* of the system. Within process theory a broad spectrum of equivalences is defined [17, 15]. An equivalence defines when two processes are equal, and hence when two different systems show the same behavior. Furthermore, a logic may *agree with* such an equivalence, meaning that two processes that are equal according to some equivalence either both satisfy a property expressed in this logic, or they both do not satisfy it. We also say that such a logic and properties expressed in this logic are *preserved* by the equivalence. Equivalences play an important role in the field of model checking, as they can be used to find a smaller process that is equivalent to the actual process of interest. This task is also called *state space reduction*. As it is known which properties are preserved by certain equivalences, it is possible to perform the model checking task using the smaller process, and still get a valid answer.

The process of a system is often modeled using either a state-based model or an event-based model. These two types of models differ in the way they store information. However, it is commonly believed that they are equally expressive. This is supported by [13, 26], which show that multiple equivalences in one type of model correspond to similar equivalences in the other type of model, by using embeddings to convert between the two types of models. Furthermore, such embeddings have practical use, as they can be used to make state space reduction algorithms for one type of model applicable to the other type as well.

1.2 Problem description

As SPL may (and often do) consist of safety critical software, there is a desire to apply model checking techniques for the verification of SPL. This can trivially be done by applying these techniques to each product separately. However, this is often computationally expensive since industrial SPL can contain very large numbers of products, and it would more or less defeat the purpose of regarding the entire set of products as a single SPL. Hence, formalisms have been proposed [10, 21] to model the process of an entire SPL at once, along with algorithms to perform model checking using such a model.

As on the single-product level, both state-based variants [9, 10] and event-based variants [5, 6] of such formalisms have been used. However, no research has been done on the relationships between these two types of models on the product-family level. Therefore it is unknown whether event-based models and state-based models on the product-family level are equally expressive, as is the common consensus on the single-product level. Hence, in this thesis we look into the following research question:

Are event-based models and state-based models on the product-family level equally expressive?

Secondly, as stated in the previous section, equivalences are powerful tools in the process of model checking, as they allow for state space reduction. Therefore it is desirable that equivalences are developed for the product-family level models, as well as methods to utilize these equivalences for the task of state space reduction. Hence, besides the defined research question, we have the following research goal:

Develop an algorithm to perform state space reduction on product-family level models.

1.3 Results and outline

In this thesis we look into several equivalences for both state-based and event-based models of SPL. The state-based model has been used by Classen et al. [10, 9] to perform product-family based verification. A different method to perform family-based verification using event-based models was proposed by Ter Beek & De Vink [5, 6]. In order to obtain some insights in the similarities and differences of these approaches, we investigate how these two types of models and the equivalences defined on them correspond to each other. For this purpose we define embeddings to convert between state-based and event-based models. We show that these embeddings preserve and reflect one of the defined equivalences.

Furthermore, we use the proposed equivalences to define minimal representations of both types of models. Subsequently, we design an algorithm that calculates one of these minimal representations for the event-based model. Using the defined embeddings, we show that state space minimization in one world does not completely correspond to minimization in the other world. In order to work around this issue, we propose small changes to the embeddings and the minimization methods. With this changes into place, we are able to show that it is possible to minimize a state-based model by minimizing the embedded artifact in the event-based world. This result indicates that event-based and state-based models may be equally expressive. Also, it proves that we can apply our state space reduction algorithm for event-based models to state-based models as well.

As a final contribution we perform a small case study to investigate the practical usefulness of the proposed state space reduction algorithm for event-based models. We find that applying this technique greatly speeds up the verification process.

The remainder of this document consists of the following chapters. We start with some preliminaries regarding the formal description of SPL in Chapter 2. Chapter 3 describes the models we will be working with. Chapter 4 defines several equivalence relations on these models, and in Chapter 5 formal definitions of minimal representations of the models are given. In Chapter 6 we describe an algorithm for reduction of one of the models, using one of the equivalences, and Chapter 7 contains theoretical results about the possibilities of model reduction using transformation to a different type of model. Lastly, we developed a toolset for performing SPL reduction and verification using mCRL2, and the results of a small case study using this toolset are presented in Chapter 8. We conclude with Chapter 9.

Chapter 2

Preliminaries

This chapter contains preliminaries on the formal descriptions of SPL, as well as on the formal descriptions of system processes. Lastly, formal description of system requirements are discussed.

2.1 A formal description of SPL

SPL are commonly modeled by *feature diagrams* (FD). In order to use such a diagram, each product is represented as a set of features, where some features are mandatory, and hence are present in all products, and others are optional, and hence represent the differences between the products of the SPL. An example of an FD is shown in Figure 2.1.

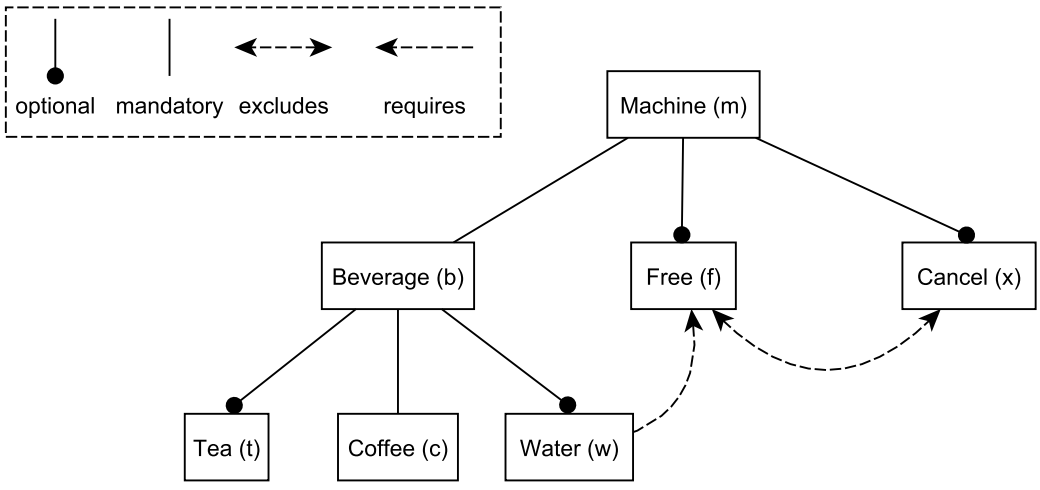


Figure 2.1: A feature diagram describing a software product line of vending machines.

We see that the FD specifies exactly which combinations of features are allowed, using different constraints. For a detailed overview of FD we refer to [27]. The example FD describes a software product line of vending machines. Each vending machine serves coffee, and may additionally serve tea and/or water. Moreover, a vending machine may serve free drinks, and it may have a cancel option. Additionally, the FD expresses that water can only be served as a free drink, and that the cancel feature is not available for machines that serve free drinks.

For the remainder of this thesis we will assume we are talking about a single SPL. For simplicity, we will abstract from the feature diagram that specifies this SPL. Instead, our SPL is specified by a tuple $(\mathcal{F}, \mathcal{P})$, where $\mathcal{F}$ is the set of features of the SPL, and $\mathcal{P} \subseteq 2^{\mathcal{F}}$ is the set of products, where each product is a subset of the set of features.

We will refer to the set $\mathbb{B}(\mathcal{F})$ of boolean expressions over the set of features as *feature expressions*. Each feature expression represents a set of products. Specifically, the semantics of a feature expression φ is the set of all products that satisfy φ.

We say a product $P \in \mathcal{P}$ satisfies a feature expression φ, denoted as $P \models \varphi$, if and only if φ evaluates to true when each boolean variable in φ corresponding to a feature that is included in P is set to true, and each boolean variable in φ corresponding to a feature that is not included in P is set to false. If P does not satisfy φ, this is denoted as $P \not\models \varphi$.

We define an equivalence relation $\sim_{\mathcal{P}}$ over the feature expressions such that two feature expressions $\varphi_1, \varphi_2 \in \mathbb{B}(\mathcal{F})$ are equivalent, denoted as $\varphi_1 \sim_{\mathcal{P}} \varphi_2$, if and only if

$$\forall P \in \mathcal{P}\colon\ (\ P \models \varphi_1 \ \Leftrightarrow\ P \models \varphi_2).$$

In a similar fashion, we say that $\varphi_1 \Rightarrow_{\mathcal{P}} \varphi_2$ if and only if

$$\forall P \in \mathcal{P}\colon\ (\ P \models \varphi_1 \ \Rightarrow\ P \models \varphi_2).$$

We will use $\hat{\varphi}$ as a shorthand notation for $[\varphi]_{\sim_{\mathcal{P}}}$, denoting the equivalence class of a feature expression φ under the equivalence $\sim_{\mathcal{P}}$. We will use $\widehat{\mathbb{B}}(\mathcal{F})$ to denote the set of all equivalence classes of feature expressions.

Lastly, consider two functions $f\colon D \to \mathbb{B}(\mathcal{F})$ and $g\colon D \to \mathbb{B}(\mathcal{F})$, for some domain D. We say that f is *stronger* than g if and only if $\forall d \in D\colon f(d) \Rightarrow_{\mathcal{P}} g(d)$.

In the next section we continue by discussing formal descriptions of system processes.

2.2 Formal descriptions of system processes

As mentioned before, thoroughly testing software is a necessary step in the development process. When working with safety critical systems it may even be considered the most important step. In order to make sure that such a system satisfies all requirements, formal descriptions can be made of both the system behavior and the requirements.

As mentioned before, the system process is the behavior of the system. A common way to describe a system process is by using process algebraic expressions, which are mathematical high-level descriptions of interactions, communications and synchronizations between multiple processes. A small example of a process algebraic description of a vending machine is shown below.

$$Machine = \sum_{b:B} receive(b) \cdot deliver(b) \cdot Machine$$

The vending machine receives an order for a certain beverage b (for instance coffee) from the set of all possible beverages B, which contains tea, coffee and water in our example from Figure 2.1. The machine delivers a cup of this beverage and starts the same process again.

A process algebraic expression in turn describes a transition system. A transition system consists of states and transitions. Each state represents a different status of the system, and the transitions represent possible changes in the system status. It is also common to describe a system process directly using a transition system, without an intermediate process algebraic description. A small example of a possible transition system for a vending machine is shown in Figure 2.2.

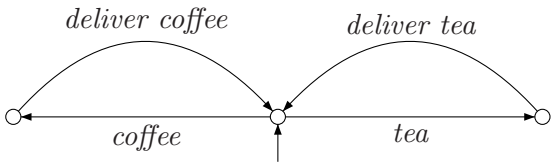


Figure 2.2: Transition system describing a vending machine serving coffee and tea.

In order to perform model checking, formal descriptions of both the system behavior and the requirements the system needs to satisfy are needed. Hence, in the next section we continue by discussing the formal description of system requirements.

2.3 Formal description of system requirements

The requirements that a system needs to satisfy are typically formalized using some logic, for instance a modal logic. We will refer to such a formalized requirement as a *property*. An example of a property in the modal μ-calculus is:

```
[tea . !deliver_tea] false
```

This property states that the situation where a `tea` action is followed by any action other than `deliver_tea` can never happen. In other words, this property states that a `tea` action (a customer orders tea from the vending machine) must always directly be followed by a `deliver_tea` action (the vending machine delivers a cup of tea to the customer), or by no action at all (the machine is unresponsive for some reason).

Regarding model checking on SPL, we say that an SPL satisfies a property if and only if all of its products satisfy this property.

Definition 2.1. *An SPL consisting of the set of products $\mathcal{P}$ satisfies a property if and only if all products in $\mathcal{P}$ satisfy this property.*

This definition immediately limits the types of logics we can use to describe the properties an SPL should satisfy. Each property should be such that we know how to verify whether a single product of the SPL satisfies this property. Whether we know how to do this depends both on the logic the property is expressed in, and on the type of model describing the process of this single product.

Chapter 3

Models

As discussed in Chapter 1, a possible way to describe a process is by using a transition system. Two common variants of transition systems are *Kripke Structures* (KS) [20] and *Labeled Transition Systems* (LTS) [19]. As both models are variants of transition systems, they both use states to represent the status of the system, and transitions to represent possible changes in system status. The models differ in the way additional information is added.

KS are so-called state-based models, implying that this information is added to the states. Each state is labeled with a set of atomic propositions, where each proposition gives some information about the current state of the system. An example of such a proposition could be `tea_ordered`. In the setting of our vending machine example, a state labeled with this proposition would represent a system status where tea has been ordered by the customer, but it is not yet delivered by the machine.

The other model, LTS, is called an event-based model. In this type of model the additional information is added to the transitions. Each transition is labeled with a single action, which indicates the event that is happening during a change of the system status. An example of an action could be `deliver_tea`. In an LTS modeling the vending machine process, a transition labeled with this action would indicate that tea is being delivered by the machine, and hence could indicate a system status change from `tea_ordered` to `idle`, for example.

As becomes clear in the above example, the semantics of state-based and event-based models are inevitably intertwined, as changes in state-information are caused by events happening in the system, and actions cause a change in system status. Therefore it could make sense to construct a model that is both state- and event-based. Such a transition system is sometimes referred to as a *Doubly Labeled Transition System* (DLTS) [14]. However, most model checking techniques are designed for either KS or LTS, and hence most tools for model checking use either KS or LTS as their underlying semantics. In practice it is therefore difficult to perform model checking on a DLTS.

In [10], *Featured Transition Systems* are introduced to represent the behavior of an entire product-family, instead of that of a single system. However, this model is actually based on DLTS, as it adds information to its states as well as to its transitions. As model checking on DLTS is difficult, the algorithms proposed by Classen et al. in [10, 9] to perform model checking on these Featured Transition Systems only make use of state information, and ignore the transition labeling. On the other hand, in [5, 6], Ter Beek & De Vink worked around

this issue by ignoring the state information in Featured Transition Systems, instead using the transition labeling. In order to be able to compare both methods, in this chapter we give formal definitions of both versions of Featured Transition Systems. That is, in Section 3.1 we present a formal definition of the pure state-based version used by Classen et al., and in Section 3.2 we give a formal definition of the pure event-based version as used by Ter Beek & De Vink. Furthermore, we discuss how these models correspond to their single-product counterparts KS and LTS, respectively. This correspondence will play a key role in the remainder of this thesis, as it allows to lift properties already proven on the single-product level to the product-family level.

3.1 State-based models

We first recall the formal definition of KS.

Definition 3.1. *A Kripke Structure is a tuple* $ks = (S, AP, \rightarrow, L, s_*)$*, where*

- S *is a finite set of states,*
- AP *is a finite set of atomic propositions,*
- $\rightarrow \subseteq S \times S$ *is the transition relation,*
- $L\colon S \rightarrow 2^{AP}$ *is the state labeling function,*
- $s_* \in S$ *is the initial state.*

We introduce some shorthand notation regarding the transition relation. If $(s, s') \in \rightarrow$, this is denoted as $s \rightarrow s'$, and we write $s \rightarrow$ as an abbreviation for $\exists s' \in S : s \rightarrow s'$.

Two KS using the same set of atomic propositions can be merged into a single KS.

Definition 3.2. *Let* $ks_i = (S_i, AP, \rightarrow_i, L_i, s_{*i})$*, for* $i \in \{1, 2\}$*, be two KS. We define the disjoint union of* ks_1 *and* ks_2*, denoted* $ks_1 \uplus ks_2$*, as the KS*

$$(S_1 \uplus S_2,\ AP,\ \rightarrow_1 \uplus \rightarrow_2,\ L_1 \uplus L_2,\ s_{*1}).$$

Now we give the definition of the state-based model for product-families as used by Classen et al.. We will refer to this model as *Feature Kripke Structure* (FKS).

Definition 3.3. *A Feature Kripke Structure (FKS) is a tuple* $fks = (S, AP, \theta, L, s_*)$*, where*

- S *is a finite set of states,*
- AP *is a finite set of atomic propositions,*
- $\theta\colon S \times S \rightarrow \mathbb{B}(\mathcal{F})$ *is the transition constraint function,*
- $L\colon S \rightarrow 2^{AP}$ *is the state labeling function,*
- $s_* \in S$ *is the initial state.*

In this definition the transition relation $\rightarrow$ as defined for KS is replaced by the transition constraint function θ. For each transition $(s,t) \in S \times S$, a product $P \in \mathcal{P}$ may satisfy the feature expression attached to this transition ($P \models \theta(s,t)$), in which case the transition (s,t) is enabled for the product P. In case P does not satisfy the feature expression, the transition is not enabled for P.

In the special case that an SPL consists of only a single product, we have that $\varphi \sim_{\mathcal{P}} \mathsf{true}$ or $\varphi \sim_{\mathcal{P}} \mathsf{false}$, for each feature expression $\varphi \in \mathbb{B}(\mathcal{F})$. The result is that each transition in an FKS expressing the behavior of this SPL is either enabled or disabled for all products. Hence, the product-family level model is in this case equivalent to a single-product level model, which is intuitive as it indeed describes the behavior of a single product.

An example FKS expressing the behavior of the software product family of vending machines from Figure 2.1 is shown in Figure 3.1.

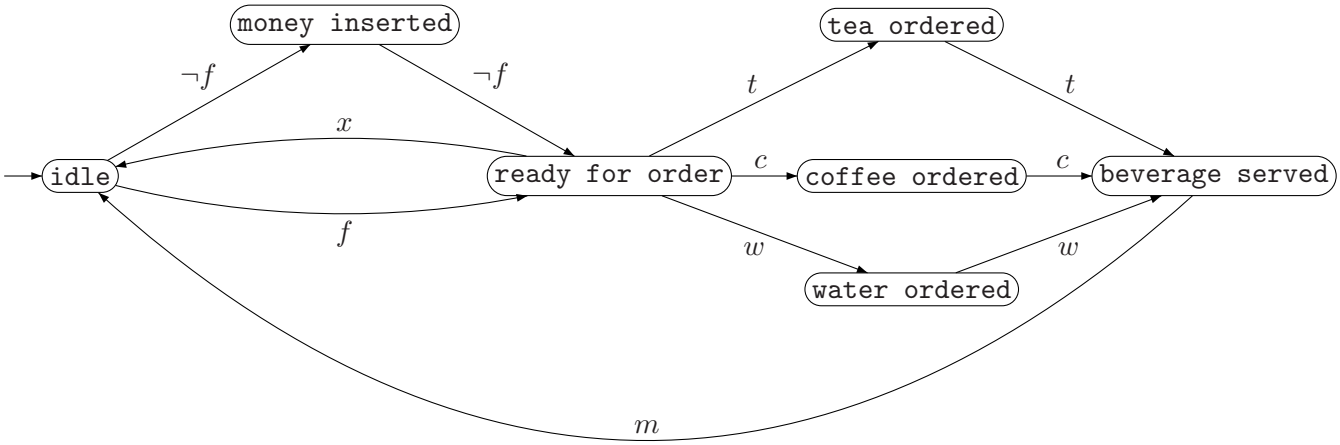


Figure 3.1: Feature Kripke Structure expressing the behavior of the vending machine SPL described by the FD from Figure 2.1. Each state is labeled with exactly one atomic proposition, which is depicted in the state. The transitions are labeled with feature expressions.

Example. The machine starts in the `idle` state (the state labeled with the atomic proposition `idle`), from where it is possible to continue to the `ready for order` state, either directly if the f(ree) feature is present, or by passing through the `money inserted` state if this feature is not present. From the `ready for order` state it is possible to go back to the `idle` state if the x (cancel) feature is present, and drinks can be ordered for which the corresponding feature is present. From any of the `beverage ordered` states the process will continue to the `beverage served` state, after which the machine returns to being `idle`.

We introduce some shorthand notation regarding the transition constraint function. If $\theta(s,s') = \psi$ this is denoted as $s \xrightarrow{\psi} s'$, and we write $s \rightarrow s'$ if $\theta(s,s') \not\sim_{\mathcal{P}} \mathsf{false}$. Similar to KS, we use $s \rightarrow$ as an abbreviation for $\exists s' \in S' : s \rightarrow s'$. Furthermore, we define a generalized version of the transition constraint function, $\breve{\theta}\colon S \times S \rightarrow \mathbb{B}(\mathcal{F})$. Given a product $P \in \mathcal{P}$, this function is constructed such that $P \models \breve{\theta}(s,t)$ if and only if P can reach state t from state s.

Definition 3.4. *Let $fks = (S,\ AP,\ \theta,\ L,\ s_*)$ be an FKS. The generalized transition constraint function $\check{\theta}\colon S \times S \to \mathbb{B}(\mathcal{F})$ for fks is the strongest function such that, for all $s, t, u \in S$:*

- $\check{\theta}(s, s) \sim_{\mathcal{P}} \mathsf{true}$
- $\check{\theta}(s, t)\ \wedge\ \theta(t, u)\ \Rightarrow_{\mathcal{P}}\ \check{\theta}(s, u)$

Using the generalized transition constraint function we define the reachability function for FKS to denote which states in an FKS are reachable for which products, from the initial state.

Definition 3.5. *Let $fks = (S,\ AP,\ \theta,\ L,\ s_*)$ be an FKS. The reachability function $\varrho\colon S \to \mathbb{B}(\mathcal{F})$ for fks is such that, for all $s \in S$:*

$$\varrho(s) = \check{\theta}(s_*, s)$$

As with KS, we can merge two FKS into a single FKS.

Definition 3.6. *Let $fks_i = (S_i,\ AP,\ \theta_i,\ L_i,\ s_{*\,i})$, for $i \in \{1, 2\}$, be two FKS. We define the disjoint union of fks_1 and fks_2, denoted $fks_1 \uplus fks_2$, as the FKS*

$$(\ S_1 \uplus S_2,\ AP,\ \theta',\ L_1 \uplus L_2,\ s_{*\,1}\ ),$$

where θ' is such that, for all $s, t \in S_1 \uplus S_2$:

$$\theta'(s,t) = \begin{cases} \theta_1(s,t) & \text{if } s,t \in S_1 \\ \theta_2(s,t) & \text{if } s,t \in S_2 \\ \mathsf{false} & \text{otherwise} \end{cases}$$

The KS of a single product of the SPL can be extracted from the FKS by *projecting* on the transitions that are enabled for this product.

Definition 3.7. *The projection of an FKS $fks = (S,\ AP,\ \theta,\ L,\ s_*)$ to a product $P \in \mathcal{P}$, denoted as $fks|_P$, is the KS $ks = (S,\ AP,\ \to,\ L,\ s_*)$, where $\to\ \subseteq S \times S$, such that:*

$$\forall (s,t) \in S \times S\colon\ (\ (s,t) \in \to\ \ \Leftrightarrow\ \ P \models \theta(s,t)\ ).$$

The KS resulting from projecting the FKS of Figure 3.1 on the product $\{m, b, c\}$ is shown in Figure 3.2.

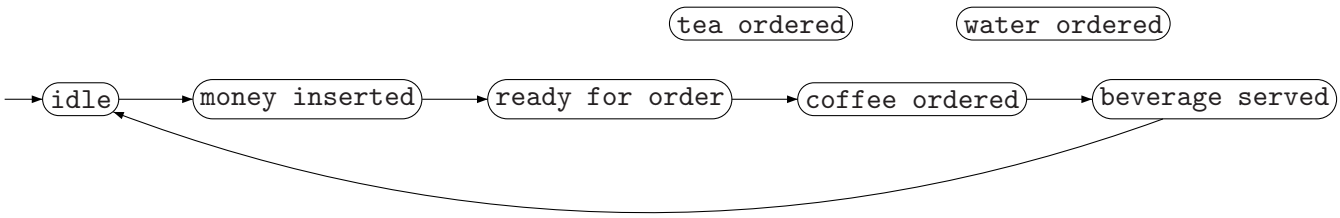


Figure 3.2: Projection of the FKS from Figure 3.1 to the product $\{m, b, c\}$.

Example. This product only consists of mandatory features, and therefore its behavior is very basic. The machine starts in the `idle` state, continues to the *money inserted* and *ready for order* states. Hereafter which it passes through the *coffee ordered* and `beverage inserted` states, and then returns to the *idle* state. All transitions associated with the t(ea), w(ater), f(ree) and x (cancel) are removed. States associated with these features are unreachable, but still present.

As an intermediary result, we prove that the projection operation distributes over disjoint union. The actual proof is omitted here, and can be found in Appendix A.

Lemma 3.1. *Let $fks_i = (S_i, AP, \theta_i, L_i, s_{*i})$, for $i \in \{1, 2\}$, be two FKS, and let $P \in \mathcal{P}$ be a product. We have:*

$$(fks_1 \uplus fks_2)|_P = fks_1|_P \uplus fks_2|_P$$

Recall Definition 2.1, stating that an SPL satisfies a certain property if and only if this property is satisfied by all its products. Hence an FKS satisfies a property if and only if the property is satisfied by its projection on each product.

Definition 3.8. *An FKS fks satisfies a property ϕ, denoted as $fks \models \phi$, iff $\forall P \in \mathcal{P}$: $fks|_P \models \phi$.*

3.2 Event-based models

We first recall the formal definition of LTS.

Definition 3.9. *A Labeled Transition System is a tuple $lts = (S, \mathcal{A}, \rightarrow, s_*)$, where*

- *S is a finite set of states,*
- *$\mathcal{A}$ is a finite set of actions,*
- *$\rightarrow \subseteq S \times (\mathcal{A} \cup \{\tau\}) \times S$ is the transition relation,*
- *$s_* \in S$ is the initial state.*

In the setting of event-based models, the special action τ is used to represent internal behavior of a system, i.e. events that are not visible by observing the system from the outside world. Transitions labeled with the action τ are also referred to as *silent transitions* or *silent steps*. The set of actions $\mathcal{A}$ is assumed not to include τ, and we use $\mathcal{A}_\tau$ as a shorthand notation for $\mathcal{A} \cup \{\tau\}$.

We also introduce some shorthand notation regarding the transition relation. If $(s, \alpha, s') \in \rightarrow$, this is denoted as $s \xrightarrow{\alpha} s'$, and we write $s \xrightarrow{\alpha}$ as an abbreviation for $\exists s' \in S : s \xrightarrow{\alpha} s'$. Two LTS using the same set of actions can be merged into a single LTS.

Definition 3.10. *Let $lts_i = (S_i, \mathcal{A}, \rightarrow_i, s_{*i})$, for $i \in \{1, 2\}$, be two LTS. We define the disjoint union of lts_1 and lts_2, denoted $lts_1 \uplus lts_2$, as the LTS*

$$(\, S_1 \uplus S_2,\ \mathcal{A},\ \rightarrow_1 \uplus \rightarrow_2,\ s_{*1}\,).$$

Now we give the definition of the event-based model for product-families as used by Ter Beek & De Vink, which we will refer to as *Feature Labeled Transition Systems* (FTS).

Definition 3.11. *A Feature Labeled Transition System is a tuple* $fts = (S, \mathcal{A}, \theta, s_*)$*, where*

- S *is a finite set of states,*
- $\mathcal{A}$ *is a finite set of actions,*
- $\theta \colon S \times \mathcal{A} \times S \to \mathbb{B}(\mathcal{F})$ *is the transition constraint function,*
- $s_* \in S$ *is the initial state.*

Similar to the state-based model, the transition relation $\rightarrow$ as defined for LTS is replaced by the transition constraint function θ. For each transition $(s, \alpha, t) \in S \times \mathcal{A}_\tau \times S$, a product $P \in \mathcal{P}$ may satisfy the feature expression attached to this transition ($P \models \theta(s, \alpha, t)$), in which case the transition (s, α, t) is enabled for the product P. In case P does not satisfy the feature expression, the transition is not enabled for P.

As with FKS, each transition is either enabled or disabled for all product in the special case that an FTS defines the behavior of an SPL consisting of a single product. Hence, the product-family level model is in this case equivalent to the single-product level model.

An example FTS expressing the behavior of the software product family of vending machines from Figure 2.1 is shown in Figure 3.3.

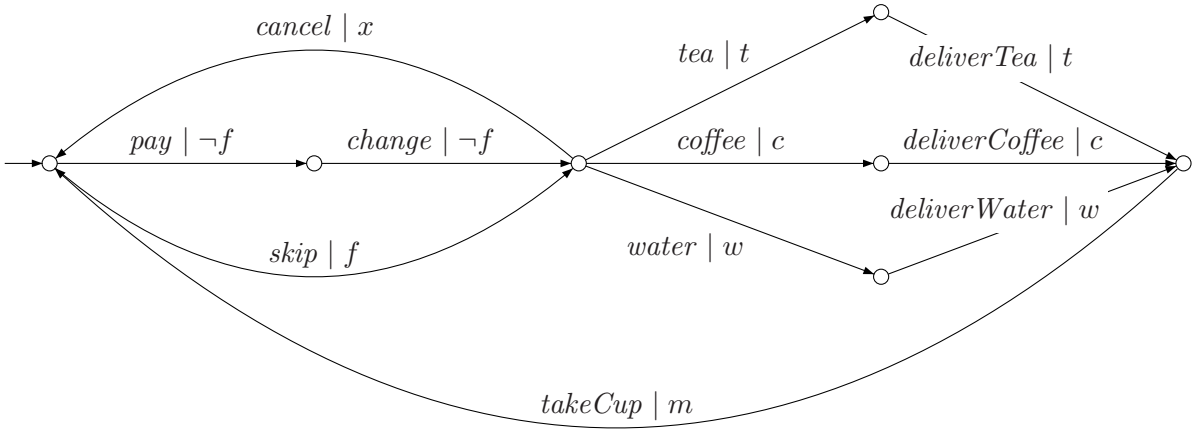


Figure 3.3: Feature labeled transition system expressing the behavior of the vending machine SPL described by the FD from Figure 2.1. The transitions are labeled with actions (left) and feature expressions (right).

Example. From the initial state it is possible to *pay* and get *change*, or to *skip* this step if the f(ree) feature is present. Then is possible to go back to the initial state if the x (cancel) feature is present, or a beverage can be ordered for which the corresponding feature is present. After ordering a beverage, the machine will *deliver* the beverage, after which the cup must be taken out of the machine to return to the initial state.

We introduce some shorthand notation regarding the transition constraint function. If $\theta(s, \alpha, s') = \psi$ this is denoted as $s \xrightarrow{\alpha|\psi} s'$, and we write $s \xrightarrow{\alpha} s'$ if $\theta(s, \alpha, s') \not\sim_{\mathcal{P}} \mathsf{false}$. As for LTS, we use $s \xrightarrow{\alpha}$ as an abbreviation for $\exists s' \in S' : s \xrightarrow{\alpha} s'$. Furthermore, we again define a generalized version of transition constraint function, $\check{\theta}\colon S \times S \to \mathbb{B}(\mathcal{F})$. Given a product $P \in \mathcal{P}$, this function is constructed such that $P \models \check{\theta}(s, t)$ if and only if P can reach state t from state s.

Definition 3.12. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS. The generalized transition constraint function $\check{\theta}\colon S \times S \to \mathbb{B}(\mathcal{F})$ for fts is the strongest function such that, for all $s, t, u \in S$ and for all $\alpha \in \mathcal{A}_\tau$:*

- $\check{\theta}(s, s) \sim_{\mathcal{P}} \mathsf{true}$
- $\check{\theta}(s, t) \wedge \theta(t, \alpha, u) \Rightarrow_{\mathcal{P}} \check{\theta}(s, u)$

Using the generalized transition constraint function we define the reachability function for FTS to denote which states in an FTS are reachable for which products, from the initial state.

Definition 3.13. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS. The reachability function $\varrho\colon S \to \mathbb{B}(\mathcal{F})$ for fts is such that, for all $s \in S$:*

$$\varrho(s) = \check{\theta}(s_*, s)$$

As with LTS, we can merge two FTS into a single FTS.

Definition 3.14. *Let $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*i})$, for $i \in \{1, 2\}$, be two FTS. We define the disjoint union of fts_1 and fts_2, denoted $fts_1 \uplus fts_2$, as the FTS*

$$(\, S_1 \uplus S_2,\ \mathcal{A},\ \theta',\ s_{*1}\,),$$

where θ' is such that, for all $s, t \in S_1 \uplus S_2$ and for all $\alpha \in \mathcal{A}_\tau$:

$$\theta'(s, \alpha, t) = \begin{cases} \theta_1(s, \alpha, t) & \text{if } s, t \in S_1 \\ \theta_2(s, \alpha, t) & \text{if } s, t \in S_2 \\ \mathsf{false} & \text{otherwise} \end{cases}$$

The LTS of a single product of the SPL can be extracted from the FTS by *projecting* on the transitions that are enabled for this product.

Definition 3.15. *The projection of an FTS $fts = (S, \mathcal{A}, \theta, s_*)$ to a product $P \in \mathcal{P}$, denoted as $fts|_P$, is the LTS $lts = (S, \mathcal{A}, \to, s_*)$, where $\to \subseteq S \times \mathcal{A} \times S$, such that:*

$$\forall t \in S \times \mathcal{A} \times S\colon t \in \to \ \Leftrightarrow\ P \models \theta(t).$$

The LTS resulting from projecting the FTS of Figure 3.3 on the product $\{m, b, c\}$ is shown in Figure 3.4.

Example. This product only consists of mandatory features, and therefore its behavior is very basic. From the initial state the actions *pay*, *change*, *coffee*, *deliverCoffee* and *takeCup* can be performed consecutively to return to the initial state. All transitions associated with the t(ea), w(ater), f(ree) and x (cancel) are removed. States associated with these features are unreachable, but still present.

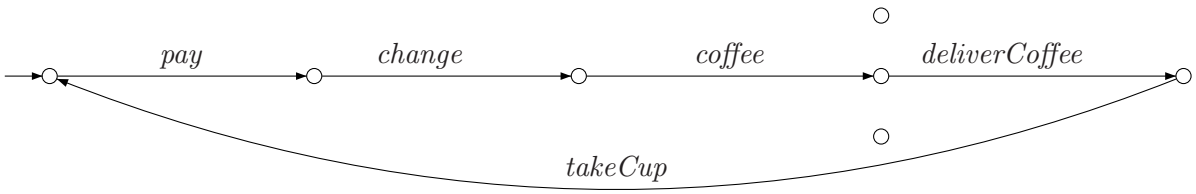


Figure 3.4: Projection of the FTS from Figure 3.3 to the product $\{m, b, c\}$.

As with state-based models, the projection operation distributes over disjoint union. Again the proof is omitted here. It can be found in Appendix A.

Lemma 3.2. *Let $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*i})$, for $i \in \{1, 2\}$, be two FTS, and let $P \in \mathcal{P}$ be a product. We have:*

$$(fts_1 \uplus fts_2)|_P = fts_1|_P \uplus fts_2|_P$$

An FTS satisfies a property if and only if the property is satisfied by its projection on each product.

Definition 3.16. *An FTS fts satisfies a property ϕ, denoted as $fts \models \phi$, iff $\forall P \in \mathcal{P}$: $fts|_P \models \phi$.*

Now that we have formally defined both state-based and event-based models to describe the behavior of product families, and how these models relate to the underlying models of each individual product, we can start to define equivalences for these models. This is what we will be doing in the next chapter. However, we would first like to point an interesting difference between the models on the product-family level and the models on single-product level.

As discussed within the first few paragraphs of this chapter, at the single-product level it is the case that all additional information in KS is added to the states, and in LTS all additional information is added to the transitions. In order to transform the models to product-family level, in both cases one extra dimension of such information was added: the feature dimension. However, in both the state-based and event-based models this extra dimension of information was added to the transitions, which is fine for the event-based model, but seems to violate the property of state-based models that all additional information should be added to the states. Although this approach is counterintuitive from a definitional point of view, for our purposes it is necessary to define the product-family level state-based model as in [10, 9].

Furthermore, the additional feature information describes which parts of the model are present for which products. In other words, it describes which parts of the model are 'enabled'. Whereas being enabled is a very intuitive notion for transitions, it is not clear what it means for a state to be enabled. Defining a semantics for this is something we leave to future work.

While keeping this discrepancy between the state-based models on single-product level and those on product-family level in mind, we continue by defining equivalences for FTS and FKS.

Chapter 4

Equivalences

As mentioned in Chapter 1, equivalence relations are widely used in model checking, mostly to perform state space reduction. In this chapter we consider two well-known equivalence relations: strong bisimulation and branching bisimulation. For both the state-based model and event-based model we will recall the original definitions of these relations, and then proceed with extending the relations to the product-family level.

4.1 Strong bisimulation

(Strong) bisimulation for processes was proposed in [24, 23]. It considers two processes to be equal if they can execute the same sequences of actions to again reach equivalent states. Strong bisimulation is the finest equivalence on Van Glabbeek's linear time - branching time spectrum [17, 15].

In this section we will propose a generalized version of bisimulation for product-family level models, for both state-based models (Section 4.1.1) and event-based models (Section 4.1.2). We investigate the relation between the newly defined equivalences and the original equivalences on the single-product level. Using the established relationship, we will lift some important properties of bisimulation from the single-product level to the product-family level.

4.1.1 State-based models

We start by giving the definition of strong bisimulation for Kripke Structures. Using this definition we define when two states of a KS are bisimilar, and when two KS are bisimilar.

Definition 4.1. *Let $ks = (S, AP, \rightarrow, L, s_*)$ be a KS. A symmetric relation R on S is called a strong bisimulation relation for KS if and only if for all states $s, t \in S$ such that $(s, t) \in R$, the following conditions are satisfied:*

1. *$L(s) = L(t)$.*
2. *If $s \rightarrow s'$, for some $s' \in S$ then there exists $t' \in S$ such that $t \rightarrow t'$ and $(s', t') \in R$.*

We say two states $s, t \in S$ are (strongly) bisimilar, denoted by $ks \models s \leftrightarrows t$, if and only if there exists a strong bisimulation relation R for KS on ks such that $(s, t) \in R$.

Let $ks' = (S', AP, \rightarrow', L', s'_*)$ be a second KS. We say ks and ks' are (strongly) bisimilar, denoted by $ks \leftrightarrows ks'$ if and only if $ks \uplus ks' \models s_* \leftrightarrows s'_*$.

In [23] it has been established that bisimilarity for KS is an equivalence relation.

Lemma 4.1. *Bisimilarity for KS ($\leftrightarrows$) is an equivalence.*

Next we adapt the definition of bisimulation of KS to be applicable in FKS. We do this by extending the bisimulation relation with feature expressions, in such a way that two states related by a feature expression φ are bisimilar for all products satisfying φ.

Definition 4.2. *Let $fks = (S, AP, \theta, L, s_*)$ be an FKS. A relation $R \subseteq S \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ is called a (strong) feature bisimulation relation for FKS if and only if R is symmetric, i.e. $(s, \hat{\varphi}, t) \in R$ iff $(t, \hat{\varphi}, s) \in R$, and if for all states $s, t \in S$ and for all feature expressions $\varphi \in \mathbb{B}(\mathcal{F})$ such that $(s, \hat{\varphi}, t) \in R$, the following conditions are satisfied:*

1. *$L(s) = L(t)$.*
2. *If $s \xrightarrow{\psi} s'$ for some $s' \in S$ and for some $\psi \in \mathbb{B}(\mathcal{F})$, then there exist states $t_1 \dots t_n \in S$ and feature expressions $\psi_1 \dots \psi_n, \varphi_1 \dots \varphi_n \in \mathbb{B}(\mathcal{F})$, for some $n \in \mathbb{N}$, such that*

$$t \xrightarrow{\psi_i} t_i \text{ and } (s', \hat{\varphi}_i, t_i) \in R,$$

for all $1 \leq i \leq n$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}}\; \bigvee_{1 \leq i \leq n} \psi_i \wedge \varphi_i.$$

The first condition of this definition is equal to the first condition of bisimulation for KS, stating that two states can only be related if they have the same label. The second condition states that if two states s and t are related for a feature expression φ, and $s \xrightarrow{\psi} s'$, then t must be able to mimic this transition for all products that satisfy φ and ψ. However, different outgoing transitions of t may be used to mimic the step for different products.

Alternatively, condition 2 could be written as

$$\forall s' \in S\colon (\, \varphi \wedge \theta(s, s') \;\Rightarrow_{\mathcal{P}}\; \bigvee \{\, \theta(t, t') \wedge \varphi' \mid (s', \hat{\varphi}', t') \in R \,\}\,).$$

Here, we do not demand a transition is actually present between s and s'; if there is none, then $\theta(s, s') \sim_{\mathcal{P}} \mathsf{false}$, and hence the condition is trivially satisfied. Similarly, we do not demand there is a transition between t and t'; if there is none, then $\theta(t, t') \sim_{\mathcal{P}} \mathsf{false}$, and hence t' simply does not contribute to satisfying the condition.

The transfer diagram for feature bisimulation for FKS is shown in Figure 4.1.

As with bisimulation for KS, we can use a feature bisimulation relation to determine when two states are feature bisimilar, and when two FKS are feature bisimilar. An example of an FKS that is feature bisimilar to that of Figure 3.1 is shown in Figure 4.2.

Example. In Figure 4.2 all vending machines with the f(ree) feature have a separate `ready for order` state, but this does not change the behavior of the system. Also note that there is no transition from the `ready for order` state for paid products to the `water ordered` state. This also does not change the behavior, since there are no products in the SPL that satisfy the feature expression $\neg f \wedge w$.

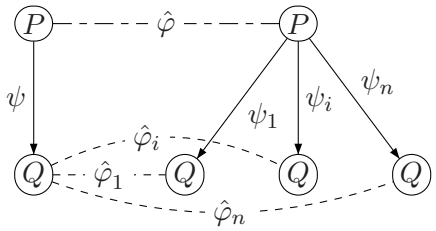


Figure 4.1: Transfer diagram for strong feature bisimulation for FKS. P and Q are sets of atomic propositions.

Definition 4.3. *Let $fks = (S, AP, \theta, L, s_*)$ be an FKS. We say two states $s, t \in S$ are (strongly) feature bisimilar for a feature expression $\varphi \in \mathbb{B}(\mathcal{F})$, denoted by $fks \models s \overset{\varphi}{\leftrightarrows}_f t$ if and only if there exists a strong feature bisimulation relation R for FKS on fks such that $(s, \hat{\varphi}, t) \in R$. Furthermore, we use $fks \models s \leftrightarrows_f t$ as shorthand notation for $fks \models s \overset{\mathsf{true}}{\leftrightarrows}_f t$.*

Let $fks' = (S', AP, \theta', L', s'_*)$ be a second FKS. We say fks and fks' are (strongly) feature bisimilar for a feature expression $\varphi \in \mathbb{B}(\mathcal{F})$, denoted by $fks \overset{\varphi}{\leftrightarrows}_f fks'$ if and only if $fks \uplus fks' \models s_* \overset{\varphi}{\leftrightarrows}_f s'_*$. Furthermore, we use $fks \leftrightarrows_f fks'$ as shorthand notation for $fks \overset{\mathsf{true}}{\leftrightarrows}_f fks'$.

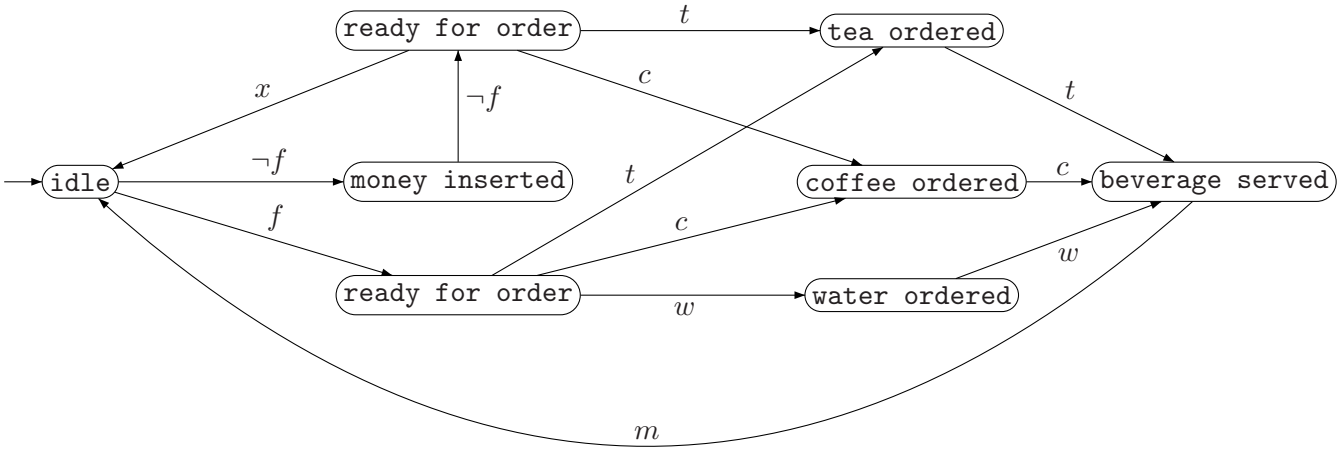


Figure 4.2: FKS that is feature bisimilar to that of Figure 3.1.

The next step is to establish how feature bisimilarity on the product-family level relates to bisimilarity on the single-product level. In order to do this we use the relation between the product-family level model and the single-product-level models, which is defined by the projection operation (Definition 3.7). Formalizing the relation between feature bisimilarity and bisimilarity will allow us to lift properties proven about bisimilarity on the single-product level to the newly defined feature bisimilarity on the product-family level. An example of such a property is Lemma 4.1, stating that bisimilarity for KS is an equivalence.

First we prove that if two states in an FKS are feature bisimilar for some feature expression ϕ, this implies that those states are bisimilar in the projected KS of all products that satisfy ϕ.

Theorem 4.1. *Let fks* $= (S,\ AP,\ \theta,\ L,\ s_*)$ *be an FKS, and let* $s_1, s_2 \in S$ *and* $\phi \in \mathbb{B}(\mathcal{F})$*. It holds that*

$$\mathit{fks} \models s_1 \overset{\phi}{\leftrightarrows}_f s_2 \ \Rightarrow\ \forall P \in \mathcal{P}\colon (P \models \phi \ \Rightarrow\ \mathit{fks}|_P \models s_1 \leftrightarrows s_2).$$

Proof. Let *fks* $= (S,\ AP,\ \theta,\ L,\ s_*)$ be an FKS, and let R be a feature bisimulation relation on *fks* such that $(s_1, \hat{\phi}, s_2) \in R$, for some $s_1, s_2 \in S$ and $\phi \in \mathbb{B}(\mathcal{F})$. Let $P \in \mathcal{P}$ be a product such that $P \models \phi$. We define the relation

$$R|_P = \{(s,t) \mid (s, \hat{\varphi}, t) \in R \wedge P \models \varphi\}.$$

We have to show that $R|_P$ is a bisimulation relation on the KS $\mathit{fks}|_P = (S,\ AP,\ \rightarrow,\ L,\ s_*)$ relating s_1 and s_2. Since $(s_1, \hat{\phi}, s_2) \in R$ and $P \models \phi$, we have that $s_1\, R|_P\, s_2$. Hence, it remains to establish that $R|_P$ is a bisimulation relation.

To show that $R|_P$ is a bisimulation relation, we first have to show that it is symmetric. Pick states $s, t \in S$ such that $(s,t) \in R|_P$. By definition of $R|_P$ this implies that $(s, \hat{\varphi}, t) \in R \wedge P \models \varphi$, for some $\varphi \in \mathbb{B}(\mathcal{F})$. Since R is symmetric we also have $(t, \hat{\varphi}, s) \in R$, and hence $(t,s) \in R|_P$, which confirms that $R|_P$ is symmetric.

Next, we have to prove that the pairs of states $(s,t) \in S \times S$ such that $(s, \hat{\varphi}, t) \in R$ and $P \models \varphi$, for some $\varphi \in \mathbb{B}(\mathcal{F})$, satisfy the conditions from Definition 4.1.

Since R is a feature bisimulation relation on *fks* and $(s, \hat{\varphi}, t) \in R$, by Definition 4.2 we have $L(s) = L(t)$, and hence the first condition is satisfied.

Suppose that $s \rightarrow s'$ for some $s' \in S$. It must be shown that

- there exists $t' \in S$ such that $t \rightarrow t'$ and $s'\, R|_P\, t'$.

By Definition 3.7 it follows that $s \xrightarrow{\psi} s'$ in *fks*, for some $\psi \in \mathbb{B}(\mathcal{F})$ such that $P \models \psi$. Since $(s, \hat{\varphi}, t) \in R$ for some $\varphi \in \mathbb{B}(\mathcal{F})$ such that $P \models \varphi$, by Definition 4.2 we have that there exist states $t_i \in S$ and feature expressions $\psi_i, \varphi_i \in \mathbb{B}(\mathcal{F})$, for $1 \leq i \leq n$, for some $n \in \mathbb{N}$, such that

$$t \xrightarrow{\psi_i} t_i \text{ and } (s', \hat{\varphi}_i, t_i) \in R,$$

for all $1 \leq i \leq n$, and such that

$$\forall P \in \mathcal{P}\colon\ P \models \varphi \wedge \psi \ \Rightarrow\ P \models \textstyle\bigvee_{1 \leq i \leq n} \psi_i \wedge \varphi_i.$$

Since $P \models \varphi \wedge \psi$ it follows that $P \models \bigvee_{1 \leq i \leq n} \psi_i \wedge \varphi_i$. We pick i such that $P \models \psi_i \wedge \varphi_i$. Since $t \xrightarrow{\psi_i} t_i$, we have that $t \rightarrow t_i$ in $\mathit{fks}|_P$, by Definition 3.7. Furthermore, since $(s', \hat{\varphi}_i, t_i) \in R$ and $P \models \varphi_i$, by definition of $R|_P$ we have that $s'\, R|_P\, t_i$.

This satisfies the second condition, confirming that $R|_P$ is indeed a bisimulation relation on $\mathit{fks}|_P$. □

As a corollary we lift this result to two FKS. The proof is included in Appendix B.

Corollary 4.1. *Let* $\mathit{fks}_i = (S_i,\ AP,\ \theta_i,\ L_i,\ s_{*\,i})$*, for* $i \in \{1,2\}$*, be two FKS, and let* $\varphi \in \mathbb{B}(\mathcal{F})$ *be a feature expression. It holds that*

$$\mathit{fks}_1 \overset{\varphi}{\leftrightarrows}_f \mathit{fks}_2 \ \Rightarrow\ \forall P \in \mathcal{P}\colon P \models \varphi \ \Rightarrow\ \mathit{fks}_1|_P \leftrightarrows \mathit{fks}_2|_P.$$

Conversely, given some FKS, we prove that if two states are bisimilar in the projected KS of all products satisfying some feature expression ϕ, this implies that those two states are feature bisimilar in the FKS for ϕ. However, this only holds for states that are labeled with the same set of atomic propositions. Without this constraint we could use this theorem to derive that every pair of states in an FKS is bisimilar for false, which is not the case, since two states that are labeled differently can never be related by a feature bisimulation relation.

Theorem 4.2. *Let fks* $= (S, AP, \theta, L, s_*)$ *be an FKS, let* $s_1, s_2 \in S$ *such that* $L(s_1) = L(s_2)$*, and let* $\phi \in \mathbb{B}(\mathcal{F})$*. It holds that*

$$(\forall P \in \mathcal{P}\colon P \models \phi \;\Rightarrow\; \mathit{fks}|_P \models s_1 \leftrightarrows s_2) \;\Rightarrow\; \mathit{fks} \models s_1 \overset{\phi}{\leftrightarrows}_f s_2.$$

Proof. Let *fks* $= (S, AP, \theta, L, s_*)$ be an FKS, let $s_1, s_2 \in S$ be two states such that $L(s_1) = L(s_2)$, and let $\phi \in \mathbb{B}(\mathcal{F})$ be a feature expression. For all products $P \in \mathcal{P}$ such that $P \models \phi$, let $R|_P$ be a bisimulation relation on $\mathit{fks}|_P$, such that $(s_1, s_2) \in R|_P$. For all products $P \in \mathcal{P}$ such that $P \not\models \phi$, let $R|_P = \emptyset$ be the empty bisimulation relation.

We define the relation

$$R = \{\, (s, \hat{\varphi}, t) \mid L(s) = L(t) \wedge \forall P \in \mathcal{P} : P \models \varphi \Leftrightarrow (s,t) \in R|_P \,\}.$$

We have to show that R is a feature bisimulation relation on *fks* such that $(s_1, \hat{\phi}, s_2) \in R$. Since $(s_1, s_2) \in R|_P \Leftrightarrow P \models \phi$, for all $P \in \mathcal{P}$, and $L(s_1) = L(s_2)$, it immediately follows that $(s_1, \hat{\phi}, s_2) \in R$, by definition of R. Hence, it remains to establish that R is a feature bisimulation relation.

To show that R is a feature bisimulation relation, we first have to show that it is symmetric. Pick states $s, t \in S$ and feature expression $\varphi \in \mathbb{B}(\mathcal{F})$ such that $(s, \hat{\varphi}, t) \in R$. By definition of R this implies that $L(s) = L(t) \wedge \forall P \in \mathcal{P} : (\, P \models \varphi \Leftrightarrow (s,t) \in R|_P \,)$. Since $R|_P$ is symmetric, for each $P \in \mathcal{P}$, we also have $L(t) = L(s) \wedge \forall P \in \mathcal{P} : (\, P \models \varphi \Leftrightarrow (t,s) \in R|_P \,)$, and hence $(t, \hat{\varphi}, s) \in R$, which confirms that R is symmetric.

Next, we have to prove that tuples $(s, \hat{\varphi}, t) \in S \times \mathbb{B}(\mathcal{F}) \times S$ such that $L(s) = L(t)$ and $\forall P \in \mathcal{P} : (\, P \models \varphi \Leftrightarrow (s,t) \in R|_P \,)$, satisfy the conditions from Definition 4.2.

By construction we have that $L(s) = L(t)$. Hence the first condition is satisfied.

Suppose that $s \xrightarrow{\psi} s'$, for some $s' \in S$ and for some $\psi \in \mathbb{B}(\mathcal{F})$. It must be shown that there exist states $t_i \in S$ and feature expressions $\psi_i, \varphi_i \in \mathbb{B}(\mathcal{F})$, for $1 \leq i \leq n$, for some $n \in \mathbb{N}$, such that

$$t \xrightarrow{\psi_i} t_i \text{ and } (s', \hat{\varphi}_i, t_i) \in R,$$

for all $1 \leq i \leq n$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}}\; \textstyle\bigvee_{1 \leq i \leq n} \psi_i \wedge \varphi_i.$$

By the definition of R it follows that for each $P \in \mathcal{P}$ such that $P \models \varphi$, we have that $(s,t) \in R|_P$. For each $P \in \mathcal{P}$ such that $P \models \psi$, it is the case that $s \to s'$ in $\mathit{fks}|_P$, by Definition 3.7. Hence, for each $P \in \mathcal{P}$ such that $P \models \varphi \wedge \psi$, by definition of bisimilarity for $R|_P$ it is the case that $t \to t_P$ in $\mathit{fks}|_P$, for some $t_P \in S$, such that $(s', t_P) \in R|_P$. By definition of projection, this means that $t \xrightarrow{\psi_P} t_P$ in *fks*, for some $\psi_P \in \mathbb{B}(\mathcal{F})$ such that $P \models \psi_P$. Furthermore,

by definition of R it follows that $(s', \hat{\varphi}_P, t_P) \in R$, for some $\varphi_P \in \mathbb{B}(\mathcal{F})$ such that $P \models \varphi_P$. Let $\mathcal{P}' = \{P \in \mathcal{P} \mid P \models \varphi \wedge \psi\}$ be the set of all products satisfying $\varphi \wedge \psi$. We conclude that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}}\; \bigvee_{P' \in \mathcal{P}'} \psi_{P'} \wedge \varphi_{P'},$$

which satisfies the second condition for feature bisimilarity, proving that relation R is indeed a feature bisimulation relation on *fks*. □

Note that it may be possible to remove the constraint of $L(s_1) = L(s_2)$ from the above theorem if feature information were to be added to the states of the FKS. By including feature information in the state labeling function it is possible to express that a state has labeling $\{a\}$ for products satisfying f, and labeling $\{b\}$ for products satisfying $\neg f$, for example. If such information were present in the FKS we could adapt the feature bisimulation requirements accordingly, by stating that two states only need to agree on the labeling for the required products in order to be feature bisimilar. This would lead to each pair of states being trivially bisimilar for the feature expression false. However, defining such a model is not in the scope of this thesis, and hence investigating it is left for future work.

We lift the result from Theorem 4.2 to two FKS. Again, the proof is included in Appendix B.

Corollary 4.2. *Let $fks_i = (S_i, AP, \theta_i, L_i, s_{*i})$, for $i \in \{1, 2\}$, be two FKS such that $L_1(s_{*1}) = L_2(s_{*2})$, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression. It holds that*

$$(\,\forall P \in \mathcal{P}\colon P \models \varphi \;\Rightarrow\; fks_1|_P \leftrightarrows fks_2|_P\,) \;\Rightarrow\; fks_1 \overset{\varphi}{\leftrightarrows}_f fks_2.$$

With Theorems 4.1 and 4.2 we have proven that being feature bisimilar on product-family level is equivalent to being bisimilar on the single-product level, under the additional constraint that states have the same labeling. As a result, it is very easy to lift results about bisimilarity on the single-product level to feature-bisimilarity on the product-family level. We start by lifting Lemma 4.1, stating that bisimilarity for KS is an equivalence, to establish that feature bisimilarity for FKS is an equivalence as well.

Theorem 4.3. *Feature bisimilarity ($\overset{\varphi}{\leftrightarrows}_f$) for FKS for given $\varphi \in \mathbb{B}(\mathcal{F})$ is an equivalence relation.*

Proof. We have to show that feature bisimilarity is reflexive, symmetric and transitive, for some feature expression $\varphi \in \mathbb{B}(\mathcal{F})$. Let $fks = (S, AP, \theta, L, s_*)$ be an FKS such that $s, t, u \in S$.

1. We prove reflexivity using s. By reflexivity of bisimilarity for KS (Lemma 4.1), we have $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fks \models s \leftrightarrows s$. Obviously $L(s) = L(s)$, and hence by Theorem 4.2 it follows that $fks \models s \overset{\varphi}{\leftrightarrows}_f s$. Hence $\overset{\varphi}{\leftrightarrows}_f$ is reflexive.

2. Suppose that $fks \models s \overset{\varphi}{\leftrightarrows}_f t$, from which it follows that $L(s) = L(t)$. By Theorem 4.1 it follows that $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fks|_P \models s \leftrightarrows t$. By symmetry of bisimilarity for KS (Lemma 4.1), we have $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fks|_P \models t \leftrightarrows s$. Since $L(t) = L(s)$, by Theorem 4.2 it follows that $fks \models t \overset{\varphi}{\leftrightarrows}_f s$. Hence $\overset{\varphi}{\leftrightarrows}_f$ is symmetric.

3. Suppose that $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$ and $\mathit{fks} \models t \overset{\varphi}{\leftrightarrows}_f u$, from which it follows that $L(s) = L(t) = L(u)$. By Theorem 4.1 it follows that $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathit{fks}|_P \models s \leftrightarrows t \wedge \mathit{fks}|_P \models t \leftrightarrows u$. By transitivity of bisimilarity for KS (Lemma 4.1), we have $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathit{fks}|_P \models s \leftrightarrows u$. Since $L(s) = L(u)$, by Theorem 4.2 it follows that $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f u$. Hence $\overset{\varphi}{\leftrightarrows}_f$ is transitive.

□

The fact that feature bisimilarity is an equivalence will come in useful when performing state space minimization, as the equivalence classes defined on an FKS by feature bisimilarity intuitively can be used as the states for the minimized model.

Related to equivalence, we can show that if states in an FKS are feature bisimilar for different feature expressions, then transitivity is preserved on a product-by-product basis.

Lemma 4.2. *Let fks be an FKS with states s, t and u such that $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$ and $\mathit{fks} \models t \overset{\psi}{\leftrightarrows}_f u$, for some $\varphi, \psi \in \mathbb{B}(\mathcal{F})$. Then $\mathit{fks} \models s \overset{\varphi \wedge \psi}{\leftrightarrows}_f u$.*

Proof. Let $\mathit{fks} = (S, AP, \theta, L, s_*)$ be an FKS with such that $s, t, u \in S$, and such that $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$ and $\mathit{fks} \models t \overset{\psi}{\leftrightarrows}_f u$, from which it follows that $L(s) = L(t) = L(u)$. By Theorem 4.1 it follows that $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathit{fks}|_P \models s \leftrightarrows t$ and $\forall P \in \mathcal{P}\colon P \models \psi \Rightarrow \mathit{fks}|_P \models t \leftrightarrows u$. By transitivity of bisimilarity for KS (Lemma 4.1), we have $\forall P \in \mathcal{P}\colon P \models \varphi \wedge \psi \Rightarrow \mathit{fks}|_P \models s \leftrightarrows u$. Since $L(s) = L(u)$, by Theorem 4.2 it follows that $\mathit{fks} \models s \overset{\varphi \wedge \psi}{\leftrightarrows}_f u$. □

Finally, we conclude that feature bisimilarity preserves the same properties for FKS as that strong bisimilarity preserves for KS.

Theorem 4.4. *A property is preserved by bisimilarity for KS if and only if this property is preserved by feature bisimilarity for FKS.*

Proof. Let fks_1 and fks_2 be two FKS such that $\mathit{fks}_1 \leftrightarrows_f \mathit{fks}_2$, and let ϕ be a property that is preserved by bisimilarity for KS. We have to show that $\mathit{fks}_1 \models \phi \Leftrightarrow \mathit{fks}_2 \models \phi$.

Assume $\mathit{fks}_1 \models \phi$. By Definition 3.8 this is equivalent to $\forall P \in \mathcal{P}\colon \mathit{fks}_1|_P \models \phi$. Since $\mathit{fks}_1 \leftrightarrows_f \mathit{fks}_2$, by Corollary 4.1 we have $\forall P \in \mathcal{P}\colon \mathit{fks}_1|_P \leftrightarrows \mathit{fks}_2|_P$. Since ϕ is preserved by $\leftrightarrows$, it follows that $\forall P \in \mathcal{P}\colon \mathit{fks}_2|_P \models \phi$, which is equivalent to $\mathit{fks}_2 \models \phi$, by Definition 3.8.

The proof for the implication in the other direction is similar.

□

The result of this last theorem in particular is very useful, as it allows to reuse all research done on logics agreeing with bisimulation for KS. In other words, if a logic at some point has been proven to be preserved by bisimilarity for KS, we can immediately conclude that this logic is also preserved by feature bisimilarity for FKS.

4.1.2 Event-based models

In this section we discuss strong bisimulation for event-based models. As the theorems in this section have proofs that are similar to the proofs of the theorems in Section 4.1.1, the proofs for these theorems are omitted here. The interested reader can find the full proofs in Appendix B.1.

As with the state-based models, we start by giving the definition of strong bisimulation in a single-product setting. For event-based models we use Labeled Transition Systems for this purpose. Using the definition of strong bisimulation we define when two states of an LTS are bisimilar, and when two LTS are bisimilar.

Definition 4.4. *Let $lts = (S, \mathcal{A}, \rightarrow, s_*)$ be an LTS. A symmetric relation R on S is called a strong bisimulation relation for LTS if and only if for all states $s, t \in S$ such that $(s,t) \in R$, the following transfer condition is satisfied:*

1. *if $s \xrightarrow{\alpha} s'$, for some $s' \in S$ and for some $\alpha \in \mathcal{A}_\tau$, then there exists $t' \in S$ such that $t \xrightarrow{\alpha} t'$ and $(s', t') \in R$.*

We say that two states $s, t \in S$ are (strongly) bisimilar, denoted by $lts \models s \leftrightarrows t$ if and only if there exists a strong bisimulation relation R for LTS on lts such that $(s,t) \in R$.

Let $lts' = (S', \mathcal{A}, \rightarrow', s'_*)$, be a second LTS. We say lts and lts' are (strongly) bisimilar, denoted by $lts \leftrightarrows lts'$ if and only if $lts \uplus lts' \models s_* \leftrightarrows s'_*$.

In [23] it has been established that bisimilarity for LTS is an equivalence relation.

Lemma 4.3. *Bisimilarity for LTS ($\leftrightarrows$) is an equivalence.*

Next we adapt the definition of bisimulation of LTS to be applicable in FTS. We do this by extending the bisimulation relation with feature expressions, similar to the approach with the state-based models. Hence, two states related by a feature expression φ are bisimilar for all products satisfying φ.

Definition 4.5. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS. A relation $R \subseteq S \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ is called a (strong) feature bisimulation relation for FTS if and only if R is symmetric, i.e. $(s, \hat{\varphi}, t) \in R$ iff $(t, \hat{\varphi}, s) \in R$, and if for all states $s, t \in S$ and for all feature expressions $\varphi \in \mathbb{B}(\mathcal{F})$ such that $(s, \hat{\varphi}, t) \in R$, the following transfer condition is satisfied:*

1. *If $s \xrightarrow{\alpha|\psi} s'$ for some $s' \in S$, $\alpha \in \mathcal{A}_\tau$ and $\psi \in \mathbb{B}(\mathcal{F})$, then there exist states $t_1 \dots t_n \in S$ and feature expressions $\psi_1 \dots \psi_n, \varphi_1 \dots \varphi_n \in \mathbb{B}(\mathcal{F})$, , for some $n \in \mathbb{N}$, such that*

$$t \xrightarrow{\alpha|\psi_i} t_i \text{ and } (s', \hat{\varphi}_i, t_i) \in R,$$

for all $1 \leq i \leq n$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}}\; \bigvee_{1 \leq i \leq n} \psi_i \wedge \varphi_i$$

The transfer condition is similar to the second condition of feature bisimulation for FKS, and states that if two states s and t are related for a feature expression φ, and $s \xrightarrow{\alpha|\psi} s'$, then t must be able to mimic this transition for a superset of all products that satisfy φ and ψ. However, different outgoing α-transitions of t can be used to mimic the step for different products. Two examples of feature bisimulation relations are shown in Figure 4.3.

Example. A superset of products is used as it allows for relations of the type shown in the left part of Figure 4.3. Such a relation is intuitively a valid feature bisimulation relation, as the two related FTS are equal. However, without the option to use supersets this relation is not valid. The tuple $(s', \hat{f}, t')$ would have to be added to the relation in order to fulfill the transfer condition.

If none of the products that satisfy φ satisfy ψ, then the transition may be mimicked by using zero paths, as illustrated in the right part of Figure 4.3.

Alternatively, the condition could be written as

$$\forall \alpha \in \mathcal{A}_\tau \forall s' \in S\colon (\, \varphi \wedge \theta(s, \alpha, s') \;\Rightarrow_{\mathcal{P}}\; \bigvee \{\, \theta(t, \alpha, t') \wedge \varphi' \mid (s', \hat{\varphi}', t') \in R \,\}\,).$$

As with bisimulation for LTS, we can use the feature bisimulation relation to determine when two states are feature bisimilar, and when two FTS are feature bisimilar.

Definition 4.6. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS. We say that two states $s, t \in S$ are (strongly) feature bisimilar for a feature expression $\varphi \in \mathbb{B}(\mathcal{F})$, denoted by $fts \models s \overset{\varphi}{\leftrightarrows}_f t$, if and only if there exists a strong feature bisimulation relation R for FTS on fts such that $(s, \hat{\varphi}, t) \in R$. Furthermore, we use $fts \models s \leftrightarrows_f t$ as shorthand notation for $fts \models s \overset{\mathsf{true}}{\leftrightarrows}_f t$.*

Let $fts' = (S', \mathcal{A}, \theta', s'_*)$ be a second FTS. We say fts and fts' are (strongly) feature bisimilar for a feature expression $\varphi \in \mathbb{B}(\mathcal{F})$, denoted by $fts \overset{\varphi}{\leftrightarrows}_f fts'$, if and only if $fts \uplus fts' \models s_* \overset{\varphi}{\leftrightarrows}_f s'_*$. Furthermore, we use $fts \leftrightarrows_f fts'$ as shorthand notation for $fts \overset{\mathsf{true}}{\leftrightarrows}_f fts'$.

As for the state-based models, we will now establish the relation between feature bisimilarity on the product-family level and the bisimilarity on the single-product level. First we prove that if two states in an FTS are feature bisimilar for some feature expression ϕ, this implies that those states are bisimilar in the projected LTS of all products that satisfy ϕ. We furthermore lift this result to two FTS.

Theorem 4.5. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, and let $s_1, s_2 \in S$ and $\phi \in \mathbb{B}(\mathcal{F})$. It holds that*

$$fts \models s_1 \overset{\phi}{\leftrightarrows}_f s_2 \;\Rightarrow\; \forall P \in \mathcal{P}\colon (P \models \phi \;\Rightarrow\; fts|_P \models s_1 \leftrightarrows s_2).$$

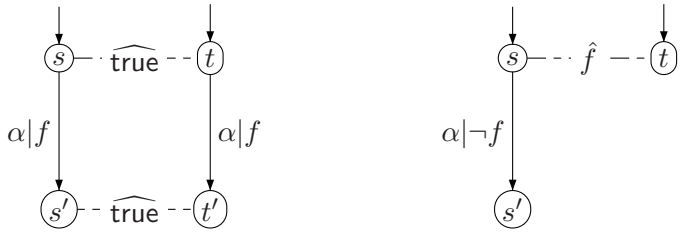


Figure 4.3: Two valid feature bisimulation relations. (Left) A relation using the option to relate supersets. (Right) A relation using the option to use zero paths.

Corollary 4.3. *Let $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*i})$, for $i \in \{1, 2\}$, be two FTS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression. It holds that*

$$fts_1 \overset{\varphi}{\leftrightarrows}_f fts_2 \;\Rightarrow\; \forall P \in \mathcal{P}\colon (\, P \models \varphi \;\Rightarrow\; fts_1|_P \leftrightarrows fts_2|_P \,)$$

Conversely, given some FTS, we prove that if two states are bisimilar in the projected LTS of all products satisfying some feature expression ϕ, this implies that those two states are feature bisimilar for ϕ in the FTS. Note that two states of an FTS are trivially related for the feature expression false, which is not true for each pair of states in an FKS, as discussed in the previous section. Hence, in this event-based setting we obtain the result that being bisimilar on the single-product level is equivalent to being feature-bisimilar on the product-family level, without any further constraints.

Theorem 4.6. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, and let $s_1, s_2 \in S$ and $\phi \in \mathbb{B}(\mathcal{F})$. It holds that*

$$(\forall P \in \mathcal{P}\colon P \models \phi \;\Rightarrow\; fts|_P \models s_1 \leftrightarrows s_2) \;\Rightarrow\; fts \models s_1 \overset{\phi}{\leftrightarrows}_f s_2.$$

We again lift this result to two FTS.

Corollary 4.4. *Let $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*i})$, for $i \in \{1, 2\}$, be two FTS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression. It holds that*

$$(\forall P \in \mathcal{P}\colon P \models \varphi \;\Rightarrow\; fts_1|_P \leftrightarrows fts_2|_P) \;\Rightarrow\; fts_1 \overset{\varphi}{\leftrightarrows}_f fts_2.$$

As in the state-based setting, we can now easily lift results obtained about bisimilarity on the single-product level to feature-bisimilarity on the product-family level. We start by lifting Lemma 4.3, stating the bisimilarity for LTS is an equivalence, to prove that feature bisimilarity for FTS is an equivalence as well.

Theorem 4.7. *Feature bisimilarity ($\overset{\varphi}{\leftrightarrows}_f$) for FTS for given $\varphi \in \mathbb{B}(\mathcal{F})$ is an equivalence relation.*

Related to equivalence, we can show that if states in an FTS are feature bisimilar for different feature expressions, then transitivity is preserved on a product-by-product basis.

Lemma 4.4. *Let fts be an FTS with states s, t and u such that $fts \models s \overset{\varphi}{\leftrightarrows}_f t$ and $fts \models t \overset{\psi}{\leftrightarrows}_f u$, for some $\varphi, \psi \in \mathbb{B}(\mathcal{F})$. Then $fts \models s \overset{\varphi \wedge \psi}{\leftrightarrows}_f u$.*

Finally, we show that feature bisimilarity preserves the same properties for FTS as that strong bisimilarity preserves for LTS. As for FKS, this particular theorem is very useful, as it eliminates the need to investigate which logics are preserved by feature bisimulation for FTS. All research done on this topic for bisimulation for LTS can immediately be used on the product-family level.

Theorem 4.8. *A property is preserved by bisimilarity for LTS if and only if this property is preserved by feature bisimilarity for FTS.*

We conclude this section by summarizing the achieved results. For both the state-based models and action-based models, we started by generalizing the definition of bisimilarity on the single-product level to a definition of feature bisimilarity on the product-family level. This resulted in Definitions 4.2, 4.3, 4.5 and 4.6. Using Theorems 4.1, 4.2, 4.5 and 4.6, we proved that two states being feature bisimilar for some feature expression φ on the product-family level is equivalent to those states being bisimilar on the single-product level of all products satisfying φ. However, for state-based models this only holds under the condition that those states are labeled with the same set of atomic propositions.

Utilizing the established correspondence between bisimilarity on the single-product level and feature bisimilarity on the product-family level we were able to show that feature bisimilarity is an equivalence, in Theorems 4.3 and 4.7. Furthermore we proved that all properties preserved by bisimilarity on the single-product level are also preserved by feature-bisimilarity on the product-family level, in Theorems 4.4 and 4.8.

In the next section we will achieve the same results for branching bisimulation.

4.2 Branching bisimulation

Branching bisimulation was proposed in [16] as an alternative to the well-known *weak bisimulation equivalence*, originally proposed by Milner as *observation equivalence* [22]. *Weak bisimulation equivalence* is a branching time equivalence for event-based models that allows to abstract from silent steps in a process. The main idea behind this concept is that an outside observer of the system cannot tell when the system performs a silent transition. He is only aware that the system is doing something if non-silent transitions are being executed. Hence, from the observer's point of view two states can be considered equivalent if they can mimic each other's steps after a series of silent transitions. This in contrast to strong bisimulation, where steps have to be mimicked immediately.

However, in [16] it was argued that this equivalence does not respect the branching structure of processes whenever such silent steps are present. This argument follows from the idea that the outside observer can not only see the current system state, but also the non-silent steps that can be performed by the system in the direct future. Hence, it is argued, two states can only be considered equivalent if they can mimic each other's steps after a series of silent transitions, *without altering the possible future behaviors during this series of silent steps.*

In order to address this problem weak bisimulation was refined to *branching bisimulation.* The corresponding equivalence in the world of state-based models is *divergence-blind stuttering equivalence*, which was first defined in [14]. It is a coarser variant of the well-known *stuttering equivalence* [8], as it ignores cycles of silent transitions (divergence).

As with strong bisimulation, we will lift these equivalences to the product-family level. We furthermore establish the relation between the equivalences on the single-product level and those on the product-family level, which we use to lift some important properties of branching bisimulation from the single-product level to the product-family level. Even though we are considering different equivalences, the proofs for the theorems in this section are very similar to those of Section 4.1, and hence they are omitted here. The interested reader is referred to Appendix B.2.

4.2.1 State-based models

We start by giving the definition of divergence-blind stuttering bisimulation for Kripke Structures. Using this definition we define when two states of a KS are divergence-blind stuttering equivalent, and when two KS are divergence-blind stuttering equivalent.

Divergence-blind stuttering equivalence is an equivalence for KS that abstracts from *silent* transitions. We say a transition $s \to s'$ between some states s and s' is silent if these states have the same label. That is, $L(s) = L(s')$. The reasoning behind this definition is that an outside observer can only observe that the system is doing something if the system status, i.e. the state labeling, changes. Therefore a transition that does not change the state labeling can not be observed, and hence is silent.

The notation $s \longrightarrow\!\!\!\!\!\rightarrow s'$ is used in KS as an abbreviation for 'a sequence of zero or more silent transitions from s to s''. More formally, $s \longrightarrow\!\!\!\!\!\rightarrow s'$ if and only if there exist states $s_0, s_1, \ldots, s_n$, such that $s_0 = s$ and $s_n = s'$, and $L(s_i) = L(s_{i+1})$ and $s_i \to s_{i+1}$, for $0 \leq i < n$, and $n \in \mathbb{N}$.

Furthermore, we use $s \dashrightarrow s'$ as an abbreviation for '$s \to s'$, or $s = s'$'.

Definition 4.7. *Let $ks = (S, AP, \to, L, s_*)$ be a KS. A symmetric relation R on S is called a divergence-blind stuttering bisimulation for KS if and only if for all states $s,t \in S$ such that $(s,t) \in R$, the following conditions are satisfied:*

1. *$L(s) = L(t)$.*
2. *If $s \to s'$, for some $s' \in S$ then there exist states $t', t'' \in S$ such that $t \longrightarrow\!\!\!\!\!\rightarrow t' \dashrightarrow t''$ and $(s,t') \in R$ and $(s',t'') \in R$.*

We say that two states $s,t \in S$ are divergence-blind stuttering equivalent, which we denote as $ks \models s \approx_{dbs} t$, if and only if there exists a divergence-blind stuttering bisimulation relation R on ks such that $(s,t) \in R$.

Let $ks' = (S', AP, \to', L', s'_*)$ be a second KS. We say ks and ks' are divergence-blind stuttering equivalent, denoted by $ks \approx_{dbs} ks'$ if and only if $ks \uplus ks' \models s_* \approx_{dbs} s'_*$.

In [14] divergence-blind stuttering equivalence is introduced as an equivalence relation.

Lemma 4.5. *Divergence-blind stuttering equivalence ($\approx_{dbs}$) is an equivalence.*

Next we adapt the definition of divergence-blind stuttering equivalence to be applicable in FKS. We do this by extending the stuttering bisimulation relation with feature expressions, in such a way that two states related by a feature expression φ are divergence-blind stuttering equivalent for all products satisfying φ.

We adapt the generalized transition constraint function from Definition 3.4 to construct the generalized *silent* transition constraint function $\check{\theta}_\tau \colon S \times S \to \mathbb{B}(\mathcal{F})$. Given a product $P \in \mathcal{P}$, this function is constructed such that $P \models \check{\theta}_\tau(s,t)$ if and only if P can reach state t from state s by only using silent transitions.

Definition 4.8. *Let $fks = (S, AP, \theta, L, s_*)$ be an FKS. The generalized silent transition constraint function $\check{\theta}_\tau \colon S \times S \to \mathbb{B}(\mathcal{F})$ for fks is the strongest function such that, for all $s,t,u \in S$:*

- $\check{\theta}_\tau(s,s) \sim_{\mathcal{P}} \mathsf{true}$
- $\check{\theta}_\tau(s,t) \wedge L(t) = L(u) \wedge \theta(t,u) \Rightarrow_{\mathcal{P}} \check{\theta}_\tau(s,u)$

The notation $s \xrightarrow{\Psi}\!\!\!\!\!\rightarrow t$ is used in FKS as a abbreviation for 'a sequence of zero or more silent transitions from s to t, with combined feature expression Ψ'. More formally, $s \xrightarrow{\Psi}\!\!\!\!\!\rightarrow t$ if and only if $\Psi \Rightarrow_{\mathcal{P}} \check{\theta}_\tau(s,t)$.

Furthermore, we use $s \overset{\psi}{\dashrightarrow} s'$ as an abbreviation for '$s \xrightarrow{\psi} s'$, or $s = s'$ and $\psi = \mathsf{true}$'.

Definition 4.9. *Let $fks = (S, AP, \theta, L, s_*)$ be an FKS. A relation $R \subseteq S \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ is called a divergence-blind stuttering feature bisimulation relation if and only if R is symmetric, i.e. $(s, \hat{\varphi}, t) \in R$ iff $(t, \hat{\varphi}, s) \in R$, and if for all states $s,t \in S$ and for all feature expressions $\varphi \in \mathbb{B}(\mathcal{F})$ such that $(s, \hat{\varphi}, t) \in R$, the following conditions are satisfied:*

1. *L(s) = L(t)*

2. *If* $s \xrightarrow{\psi} s'$ *for some* $s' \in S$ *and* $\psi \in \mathbb{B}(\mathcal{F})$*, then there exist states* $t'_1 \dots t'_n, t''_1 \dots t''_n \in S$ *and feature expressions* $\Psi_1 \dots \Psi_n, \psi_1 \dots \psi_n, \varphi_1 \dots \varphi_n, \varphi'_1 \dots \varphi'_n \in \mathbb{B}(\mathcal{F})$*, for some* $n \in \mathbb{N}$*, such that*

$$t \xrightarrow{\Psi_i}\!\!\!\!\rightarrow\!\!\!\!\rightarrow t'_i \overset{\psi_i}{\dashrightarrow} t''_i \text{ and } (s, \hat{\varphi}_i, t'_i) \in R \text{ and } (s', \hat{\varphi}'_i, t''_i) \in R,$$

for all $1 \leq i \leq n$*, and such that*

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}}\; \bigvee_{1 \leq i \leq n} \Psi_i \wedge \varphi_i \wedge \psi_i \wedge \varphi'_i$$

The second condition of this definition differs from that of feature bisimulation for FKS in the sense that a step does not have to be mimicked immediately. Instead, a number of silent steps may be present before mimicking the actual transition. Furthermore, the transition can be mimicked using zero steps if the transition was silent.

Alternatively, condition 2 could be written as

$$\forall s' \in S\colon\; \varphi \wedge \theta(s, s') \;\Rightarrow_{\mathcal{P}}$$

$$\bigvee \{\, \check{\theta}_\tau(t, t') \wedge (t' = t'' \vee \theta(t', t'')) \wedge \varphi' \wedge \varphi'' \mid (s, \hat{\varphi}', t'), (s', \hat{\varphi}'', t'') \in R \,\}\,.$$

We use a divergence-blind stuttering feature bisimulation relation to determine when two states are divergence-blind stuttering feature equivalent, and when two FKS are divergence-blind stuttering feature equivalent. An example of two divergence-blind stuttering feature equivalent FKS is shown in Figure 4.4.

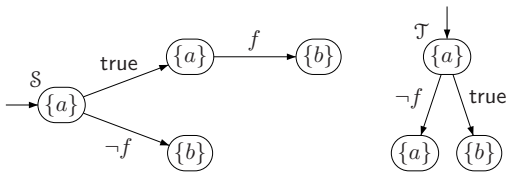


Figure 4.4: Two divergence-blind stuttering feature equivalent FKS.

Definition 4.10. *Let* $fks = (S, AP, \theta, L, s_*)$ *be an FKS. We say that two states* $s, t \in S$ *are divergence-blind stuttering feature equivalent for a feature expression* $\varphi \in \mathbb{B}(\mathcal{F})$*, denoted by* $fks \models s \overset{\varphi}{\approx}_{dbsf} t$ *if and only if there exists a divergence-blind stuttering feature bisimulation relation* R *on* fks *such that* $(s, \hat{\varphi}, t) \in R$*. Furthermore, we use* $fks \models s \approx_{dbsf} t$ *as shorthand notation for* $fks \models s \overset{\mathsf{true}}{\approx}_{dbsf} t$*.*

Let $fks' = (S', AP, \theta', L', s'_*)$ *be a second FKS. We say* fks *and* fks' *are divergence-blind stuttering feature equivalent for a feature expression* $\varphi \in \mathbb{B}(\mathcal{F})$*, denoted by* $fks \overset{\varphi}{\approx}_{dbsf} fks'$ *if and only if* $fks \uplus fks' \models s_* \overset{\varphi}{\approx}_{dbsf} s'_*$*. Furthermore, we use* $fks \approx_{dbsf} fks'$ *as shorthand notation for* $fks \overset{\mathsf{true}}{\approx}_{dbsf} fks'$*.*

As with strong bisimulation, we continue by establishing how divergence-blind stuttering feature equivalence on the product-family level relates to divergence-blind stuttering equivalence on the single-product level. We first prove that if two states in an FKS are divergence-blind stuttering feature equivalent for some feature expression ϕ, this implies that those states are divergence-blind stuttering equivalent in the KS of all products that satisfy ϕ.

Theorem 4.9. *Let* $fks = (S, AP, \theta, L, s_*)$ *be an FKS, and let* $s_1, s_2 \in S$ *and* $\phi \in \mathbb{B}(\mathcal{F})$*. It holds that*

$$fks \models s_1 \overset{\phi}{\approx}_{dbsf} s_2 \;\Rightarrow\; \forall P \in \mathcal{P}\colon (P \models \phi \;\Rightarrow\; fks|_P \models s_1 \approx_{dbs} s_2).$$

We furthermore lift this result to two FKS.

Corollary 4.5. *Let* $fks_i = (S_i, AP, \theta_i, L_i, s_{*i})$*, for* $i \in \{1, 2\}$*, be two FKS, and let* $\varphi \in \mathbb{B}(\mathcal{F})$ *be a feature expression. It holds that*

$$fks_1 \overset{\varphi}{\approx}_{dbsf} fks_2 \;\Rightarrow\; (\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fks_1|_P \approx_{dbs} fks_2|_P\,).$$

Conversely, given some FKS, we prove that if two states are divergence-blind stuttering equivalent in the KS of all products satisfying some feature expression ϕ, this implies that those two states are divergence-blind stuttering feature equivalent in the FKS for ϕ. However, as with feature bisimulation for FKS, this only holds for states that are labeled with the same set of atomic propositions.

Theorem 4.10. *Let* $fks = (S, AP, \theta, L, s_*)$ *be an FKS, and let* $s_1, s_2 \in S$ *such that* $L(s_1) = L(s_2)$*, and let* $\phi \in \mathbb{B}(\mathcal{F})$*. It holds that*

$$(\forall P \in \mathcal{P}\colon P \models \phi \;\Rightarrow\; fks|_P \models s_1 \approx_{dbs} s_2) \;\Rightarrow\; fks \models s_1 \overset{\phi}{\approx}_{dbsf} s_2.$$

We again lift this result to two FKS.

Corollary 4.6. *Let* $fks_i = (S_i, AP, \theta_i, L_i, s_{*i})$*, for* $i \in \{1, 2\}$*, be two FKS such that* $L_1(s_{*1}) = L_2(s_{*2})$*, and let* $\varphi \in \mathbb{B}(\mathcal{F})$ *be a feature expression. It holds that*

$$(\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fks_1|_P \approx_{dbs} fks_2|_P\,) \;\Rightarrow\; fks_1 \overset{\varphi}{\approx}_{dbsf} fks_2.$$

As with strong bisimulation, we are now able to easily lift properties from the single-product level models to the product-family level models. We first prove that divergence-blind stuttering feature equivalence is an equivalence relation using that divergence-blind stuttering equivalence for KS is an equivalence relation.

Theorem 4.11. *Divergence-blind stuttering feature equivalence (*$\overset{\varphi}{\approx}_{dbsf}$*) for given* $\varphi \in \mathbb{B}(\mathcal{F})$ *is an equivalence relation.*

Related to equivalence, we can show that if states in an FKS are divergence-blind stuttering feature equivalent for different feature expressions, then transitivity is preserved on a product-by-product basis.

Lemma 4.6. *Let fks be an FKS with states* s, t *and* u *such that* $fks \models s \overset{\varphi}{\approx}_{dbsf} t$ *and* $fks \models t \overset{\psi}{\approx}_{dbsf} u$*, for some* $\varphi, \psi \in \mathbb{B}(\mathcal{F})$*. Then* $fks \models s \overset{\varphi \wedge \psi}{\approx}_{dbsf} u$*.*

Finally, we conclude with the important property that divergence-blind stuttering feature equivalence preserves the same properties for FKS as that divergence-blind stuttering equivalence preserves for KS.

Theorem 4.12. *A property is preserved by divergence-blind stuttering equivalence for KS if and only if this property is preserved by divergence-blind stuttering feature equivalence for FKS.*

Similarly to strong bisimulation, the result of this theorem is particularly valuable, as it allows to reuse all research done on logics agreeing with divergence-blind stuttering equivalence for KS.

We continue with branching-bisimilarity for event-based models.

4.2.2 Event-based models

As with the state-based models, we start by giving the definition of branching bisimulation for LTS. Using this definition we define when two states of an LTS are branching bisimilar, and when two LTS are branching bisimilar.

The notation $s \longrightarrow\!\!\!\!\!\rightarrow s'$ is used in LTS as an abbreviation for 'a sequence of zero or more τ-transitions from s to s''. More formally, $s \longrightarrow\!\!\!\!\!\rightarrow s'$ if and only if there exist states $s_0, s_1, \ldots, s_n$, such that $s_0 = s$ and $s_n = s'$, and $s_i \xrightarrow{\tau} s_{i+1}$, for $0 \leq i < n$, and $n \in \mathbb{N}$.

Furthermore, we use $s \xrightarrow{(\alpha)} s'$ as an abbreviation for '$s \xrightarrow{\alpha} s'$, or $\alpha = \tau$ and $s = s'$'.

Definition 4.11. *Let $lts = (S, \mathcal{A}, \rightarrow, s_*)$ be an LTS. A symmetric relation R on S is called a branching bisimulation relation for LTS if and only if for all states $s, t \in S$ such that $(s,t) \in R$, the following transfer condition is satisfied:*

1. *if $s \xrightarrow{\alpha} s'$, for some $s' \in S$ and for some $\alpha \in \mathcal{A}_\tau$ then there exist $t', t'' \in S$ such that $t \longrightarrow\!\!\!\!\!\rightarrow t' \xrightarrow{(\alpha)} t''$ and $(s,t') \in R$ and $(s',t'') \in R$.*

We say two states $s, t \in S$ are branching bisimilar, denoted by $lts \models s \leftrightarrows_b t$ if and only if there exists a branching bisimulation relation R for LTS on lts such that $(s,t) \in R$.

Let $lts' = (S', \mathcal{A}, \rightarrow', s'_*)$ be a second LTS. We say lts and lts' are branching bisimilar, denoted by $lts \leftrightarrows_b lts'$ if and only if $lts \uplus lts' \models s_* \leftrightarrows_b s'_*$.

The transfer diagram for branching bisimulation is shown in Figure 4.5.

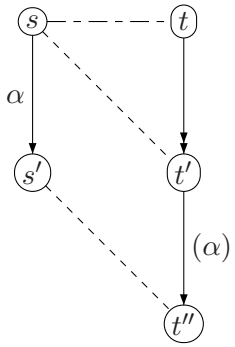


Figure 4.5: Transfer diagram for branching bisimilarity.

In [4] it has been established that branching bisimulation for LTS is an equivalence relation.

Lemma 4.7. *Branching bisimulation for LTS ($\leftrightarrows_b$) is an equivalence.*

Next we adapt the definition of branching bisimulation for LTS to be applicable in FTS. We do this by extending branching bisimulation relations with feature expressions, similar to the approach with state-based models. Hence, two states related by a feature expression φ are branching bisimilar for all products satisfying φ.

As for state-based models, we adapt the generalized transition constraint function from Definition 3.12 to construct the generalized *silent* transition constraint function $\check{\theta}_\tau \colon S \times S \to \mathbb{B}(\mathcal{F})$. Given a product $P \in \mathcal{P}$, this function is constructed such that $P \models \check{\theta}_\tau(s,t)$ if and only if P can reach state t from state s by only using silent transitions.

Definition 4.12. *Let fts $= (S, \mathcal{A}, \theta, s_*)$ be an FTS. The generalized silent transition constraint function $\check{\theta}_\tau \colon S \times S \to \mathbb{B}(\mathcal{F})$ for fts is the strongest function such that, for all $s,t,u \in S$:*

- $\check{\theta}_\tau(s,s) \sim_{\mathcal{P}} \mathsf{true}$
- $\check{\theta}_\tau(s,t) \wedge \theta(t,\tau,u) \Rightarrow_{\mathcal{P}} \check{\theta}_\tau(s,u)$

The notation $s \xrightarrow{\Psi}\!\!\!\!\!\rightarrow t$ is used in FTS as an abbreviation for 'a sequence of zero or more silenct (τ) transitions from s to t, with combined feature expression Ψ'. More formally, $s \xrightarrow{\Psi}\!\!\!\!\!\rightarrow t$ if and only if $\Psi \Rightarrow_{\mathcal{P}} \check{\theta}_\tau(s,t)$. Furthermore, we use $s \xrightarrow{(\alpha|\psi)} s'$ as an abbreviation for '$s \xrightarrow{\alpha|\psi} s'$, or $\alpha = \tau$ and $s = s'$ and $\psi = \mathsf{true}$'.

Definition 4.13. *Let fts $= (S, \mathcal{A}, \theta, s_*)$ be an FTS. A relation $R \subseteq S \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ is called a branching feature bisimulation relation if and only if R is symmetric, i.e. $(s,\hat{\varphi},t) \in R \Leftrightarrow (t,\hat{\varphi},s) \in R$, and if for all states $s,t \in S$ and for all feature expressions $\varphi \in \mathbb{B}(\mathcal{F})$ such that $(s,\hat{\varphi},t) \in R$, the following transfer condition is satisfied:*

1. *If $s \xrightarrow{\alpha|\psi} s'$ for some $s' \in S$, $\alpha \in \mathcal{A}_\tau$, and $\psi \in \mathbb{B}(\mathcal{F})$, then there exist states $t'_1 \ldots t'_n, t''_1 \ldots t''_n \in S$ and feature expressions $\Psi_1 \ldots \Psi_n, \psi_1 \ldots \psi_n, \varphi_1 \ldots \varphi_n, \varphi'_1 \ldots \varphi'_n \in \mathbb{B}(\mathcal{F})$, for some $n \in \mathbb{N}$, such that*

$$t \xrightarrow{\Psi_i}\!\!\!\!\!\rightarrow t'_i \xrightarrow{(\alpha|\psi_i)} t''_i \text{ and } (s,\hat{\varphi}_i,t'_i) \in R \text{ and } (s',\hat{\varphi}'_i,t''_i) \in R,$$

for all $1 \le i \le n$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}}\; \bigvee_{1 \le i \le n} \Psi_i \wedge \varphi_i \wedge \psi_i \wedge \varphi'_i$$

The transfer diagram for branching feature bisimulation is shown in Figure 4.6.

The transfer condition is similar to the second condition of divergence-blind feature stuttering bisimulation for FKS, adding the option to mimic steps after performing sequence of silent transitions.

Alternatively, the transfer condition could be written as

$$\forall \alpha \in \mathcal{A}_\tau \forall s' \in S \colon\; \varphi \wedge \theta(s,\alpha,s') \;\Rightarrow_{\mathcal{P}}$$

$$\bigvee \{\, \check{\theta}_\tau(t,t') \wedge (\,(t' = t'' \wedge \alpha = \tau) \vee \theta(t',\alpha,t'')\,) \wedge \varphi' \wedge \varphi'' \mid (s,\hat{\varphi}',t'),(s',\hat{\varphi}'',t'') \in R \,\}.$$

We use a branching feature bisimulation relation to determine when two states are branching feature bisimilar, and when two FTS are branching feature bisimilar.

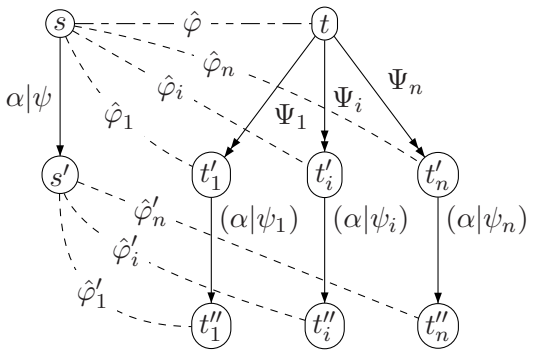


Figure 4.6: Transfer diagram for branching feature bisimilarity.

Definition 4.14. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS. We say two states $s, t \in S$ are branching feature bisimilar for a feature expression $\varphi \in \mathbb{B}(\mathcal{F})$, denoted by $fts \models s \overset{\varphi}{\leftrightarrows}_{bf} t$, if and only if there exists a branching feature bisimulation relation R on fts such that $(s, \hat{\varphi}, t) \in R$. Furthermore, we use $fts \models s \leftrightarrows_{bf} t$ as shorthand notation for $fts \models s \overset{\mathsf{true}}{\leftrightarrows}_{bf} t$.*

Let $fts' = (S', \mathcal{A}, \theta', s'_*)$ be a second FTS. We say fts and fts' are branching feature bisimilar for a feature expression $\varphi \in \mathbb{B}(\mathcal{F})$, denoted by $fts \overset{\varphi}{\leftrightarrows}_{bf} fts'$ if and only if $fts \uplus fts' \models s_* \overset{\varphi}{\leftrightarrows}_{bf} s'_*$. Furthermore, we use $fts \leftrightarrows_{bf} fts'$ as shorthand notation for $fts \overset{\mathsf{true}}{\leftrightarrows}_{bf} fts'$.

Next we prove that if two states in an FTS are branching feature bisimilar for some feature expression ϕ, this implies that those states are branching bisimilar in the LTS of all products that satisfy ϕ.

Theorem 4.13. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, and let $s_1, s_2 \in S$ and $\phi \in \mathbb{B}(\mathcal{F})$. It holds that*

$$fts \models s_1 \overset{\phi}{\leftrightarrows}_{bf} s_2 \;\Rightarrow\; \forall P \in \mathcal{P}\colon (P \models \phi \;\Rightarrow\; fts|_P \models s_1 \leftrightarrows_b s_2).$$

We furthermore lift this result to two FTS.

Corollary 4.7. *Let $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*i})$, for $i \in \{1, 2\}$, be two FTS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression. It holds that*

$$fts_1 \overset{\varphi}{\leftrightarrows}_{bf} fts_2 \;\Rightarrow\; (\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts_1|_P \leftrightarrows_b fts_2|_P).$$

Conversely, given some FTS, we prove that if two states are branching bisimilar in the LTS of all products satisfying some feature expression ϕ, this implies that those two states are divergence-blind stuttering feature equivalent in the FTS for ϕ.

Theorem 4.14. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, let $s_1, s_2 \in S$, and let $\phi \in \mathbb{B}(\mathcal{F})$. It holds that*

$$(\forall P \in \mathcal{P}\colon P \models \phi \;\Rightarrow\; fts|_P \models s_1 \leftrightarrows_b s_2) \;\Rightarrow\; fts \models s_1 \overset{\phi}{\leftrightarrows}_{bf} s_2.$$

We again lift this result to two FTS.

Corollary 4.8. *Let $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*i})$, for $i \in \{1, 2\}$, be two FTS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression. It holds that*

$$(\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts_1|_P \leftrightarrows_b fts_2|_P) \Rightarrow fts_1 \overset{\varphi}{\leftrightarrows}_{bf} fts_2.$$

We now lift Lemma 4.7 from the single-product level to the product-family level, in order to prove that branching feature bisimilarity is an equivalence relation.

Theorem 4.15. *Branching feature bisimilarity ($\overset{\varphi}{\leftrightarrows}_{bf}$) for given $\varphi \in \mathbb{B}(\mathcal{F})$ is an equivalence relation.*

Related to equivalence, we can show that if states in an FTS are branching feature bisimilar for different feature expressions, then transitivity is preserved on a product-by-product basis.

Lemma 4.8. *Let fts be an FTS with states s, t and u such that $fts \models s \overset{\varphi}{\leftrightarrows}_{bf} t$ and $fts \models t \overset{\psi}{\leftrightarrows}_{bf} u$, for some $\varphi, \psi \in \mathbb{B}(\mathcal{F})$. Then $fts \models s \overset{\varphi \wedge \psi}{\leftrightarrows}_{bf} u$.*

Finally, we conclude that branching feature bisimulation preserves the same properties for FTS as that branching bisimulation preserves for LTS.

Theorem 4.16. *A property is preserved by branching bisimulation for LTS if and only if this property is preserved by branching feature bisimulation for FTS.*

We conclude this section by summarizing the achieved results. As stated at the end of Section 4.1, we have achieved the same results for branching bisimilarity as we did for strong bisimilarity.

We started by generalizing the definitions of divergence-blind stuttering equivalence and branching bisimilarity on the single-product level to definitions of divergenge-blind stuttering feature equivalence and branching feature bisimilarity on the product-family level. This resulted in Definitions 4.9, 4.10, 4.13 and 4.14. Using Theorems 4.9, 4.10, 4.13 and 4.14, we proved that two states being divergenge-blind stuttering feature equivalent or branching feature bisimilar for some feature expression φ on the product-family level is equivalent to those states being divergenge-blind stuttering equivalent or branching bisimilar on the single-product level of all products satisfying φ. However, for state-based models this only holds under the condition that those states are labeled with the same set of atomic propositions.

Utilizing the established correspondence of the equivalences on the single-product level and those on the product-family level we were able to show that the defined relations on product-family are an equivalence, in Theorems 4.11 and 4.15. Furthermore we proved that all properties preserved by the equivalences on the single-product level are also preserved by the corresponding equivalences on the product-family level, in Theorems 4.12 and 4.16.

In the next section we relate the work done in this chapter to the work done in [11] on equivalences for state-based Featured Transition Systems.

4.3 Related work

In [11], work is presented by Cordy et al. on simulation equivalence for Featured Transition Systems. Simulation equivalence is very similar to bisimilarity, as it uses the same transfer conditions. The difference between the two definitions is rooted in symmetry. Two states are bisimilar if there exists a *symmetrical* simulation relation - a bisimulation relation - relating the two states. On the other hand, two states s and t are simulation equivalent if there exist two simulation relations, of which one relates s to t and the other relates t to s. As a bisimulation relation can act as both of these simulation relations, we find that simulation equivalence is coarser than bisimilarity.

The Featured Transition Systems used in [11] have labeled states and unlabeled transitions, making them equivalent to our FKS. Cordy et al. extended the definition of simulation equivalence for KS to featured simulation equivalence, in a very similar way as we have extended the definitions of strong bisimulation and divergence-blind stuttering equivalence to product-family based definitions. That is, the transfer conditions are formulated differently, but the idea of being able to mimic a transition using different paths for different products is identical.

However, the featured simulation relation proposed by Cordy et al. does not consist of triples of the form $S \times \mathbb{B}(\mathcal{F}) \times S$, with S a set of states, as is the case with our definition of a featured bisimulation relation. Instead, they have defined a featured simulation relation as a binary function from pairs of states to feature expressions. The consequence of this alternate formulation is that all pairs of states are trivially related for the feature expression false, regardless of the labeling of these states. At first sight this seems like an advantage, since it allows to prove that featured simulation equivalence for some feature expressions is equivalent to simulation equivalence for all product satisfying this expression. This in contrast to our results, where this is only true under the additional constraint that the state-labeling is equal. However, one of the purposes of calculating a featured (bi)simulation is to be able to perform state space reduction. Since each state has only one label, it is not possible to merge states that are labeled differently. Hence, using the approach of Cordy et al. merging states related for the feature expression false can only be done after performing an additional check of the state labelings, whereas this problem is non-existent using our formulation.

In the next chapter we will discuss state space reduction of both FTS and FKS using the equivalences we proposed in this chapter. It will then become clear under which circumstances it is desirable to merge states that are related for the feature expression false.

Chapter 5

Quotients

Now that we have defined some equivalence relations on the different models, we can define minimal representations of the models with respect to those relations. We refer to such minimal representations as *quotients*. As already discussed in Chapter 1, quotients play an important role in the field of model checking, as they can speed up the model-checking process significantly without invalidating any results.

In this chapter we will propose two types of quotients for both state-based and event-based models on the product-family level. Both types of quotients are aimed at minimizing the number of states of the model, while making sure the number of transitions does not grow.

We will refer to the first type of quotient as *naive*. This type is a straightforward generalization of the quotients at single-product level. However, it is not very powerful. Secondly, we define the *coherent* quotient. This quotient exploits the reachability information of the product-family level model, and as a result it is more powerful than the naive type. As a drawback, a single model may have multiple coherent quotients, and hence it is harder to calculate.

In Section 5.1 we discuss both types of quotients for state-based models, and in Section 5.2 we do the same for event-based models.

We will use the notation $[s]_\simeq$ to denote the equivalence class of a state $s \in S$ under some equivalence relation $\simeq$, where S is the state space of some transition system $\mathcal{S}$. That is:

$$[s]_\simeq = \{t \in S \mid \mathcal{S} \models s \simeq t\}.$$

We will use the notation $[S]_\simeq = \bigcup_{s \in S} [s]_\simeq$ to denote the set of all equivalence classes of S under $\simeq$. Note that $[S]_\simeq$ is a partition of S. We will refer to elements $C \in [S]_\simeq$ as *classes*. Complementary, given any partition P of S, we let $\simeq_P$ be the equivalence relation on S such that $[S]_{\simeq_P} = P$.

Furthermore, in this chapter we will define relations $\sim$ that are reflexive and symmetric, but not transitive. We will use the notation $\langle S \rangle_\sim$ to denote the set of maximal cliques in the graph induced on S by $\sim$. Formally,

$$\langle S \rangle_\sim = \{\ C \in 2^S \mid \forall s,t \in C \colon \mathcal{S} \models s \sim t \ \wedge\ \neg \exists C' \in 2^S \colon (C \subset C' \wedge \forall s,t \in C' \colon \mathcal{S} \models s \sim t)\ \}.$$

Note that, since $\sim$ is not transitive, $\langle S \rangle_\sim$ may not be a partition of S. That is, there may be some states in S that are included in multiple classes of $\langle S \rangle_\sim$. We still refer to elements $C \in \langle S \rangle_\sim$ as classes, even though these elements are not technically equivalence classes.

5.1 State-based models

We start by giving a definition of the bisimulation and divergent-blind stuttering quotients for KS.

Definition 5.1. *Let $ks = (S, AP, \to, L, s_*)$ be a KS, and let $\simeq \in \{\leftrightarroweq, \approx_{dbs}\}$. The $\simeq$-quotient of ks is the KS $ks' = (S', AP, \to', L', s'_*)$, such that*

- $S' = [S]_\simeq$.
- *If $\simeq = \leftrightarroweq$, we have*

$$\to' = \{\, (C_1, C_2) \in S' \times S' \mid \exists (s,t) \in C_1 \times C_2 \colon (s,t) \in \to \,\}.$$

  Otherwise, if $\simeq = \approx_{dbs}$, we have

$$\to' = \{\, (C_1, C_2) \in S' \times S' \mid C_1 \neq C_2 \wedge \exists (s,t) \in C_1 \times C_2 \colon (s,t) \in \to \,\}.$$

- $L'([s]_\simeq) = L(s)$, *for all* $s \in S$.
- $s'_* = [s_*]_\simeq$.

The function that yields the $\simeq$-quotient of a KS is denoted by $\simeq$-$\min_{\mathtt{KS}}$, for $\simeq \in \{\leftrightarroweq, \approx_{dbs}\}$. We say that a KS ks is *minimal modulo bisimulation* if and only if $\leftrightarroweq$ -$\min_{\mathtt{KS}}(ks) = ks$. Similarly, we say it is *minimal modulo divergence-blind stuttering equivalence* if and only if $\approx_{dbs}$-$\min_{\mathtt{KS}}(ks) = ks$.

To extend this definition to FKS we need to consider the extra dimension of feature expressions. In particular, we have seen that both feature bisimulation and divergence-blind stuttering feature equivalence are an equivalence for a fixed feature expression (Theorem 4.3 and Theorem 4.11). Hence we can define feature bisimulation and divergence-blind stuttering feature quotients for FKS using a fixed feature expression, in a similar way as bisimulation and divergence-blind stuttering quotients for KS are defined. We will refer to these quotients as the *naive feature bisimulation quotient* and the *naive divergence-blind stuttering feature quotient* for FKS.

Definition 5.2. *Let $fks = (S, AP, \theta, L, s_*)$ be an FKS, let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression, and let $\simeq \in \{\leftrightarroweq_f, \approx_{dbsf}\}$. The naive $\simeq$-quotient of fks for φ is the FKS $fks' = (S', AP, \theta', L', s'_*)$, such that*

- $S' = [S]_{\underline{\leftrightarroweq}^{\varphi}}$.
- $\theta' \colon S' \times S' \to \mathbb{B}(\mathcal{F})$ *is constructed such that, for all* $C_1, C_2 \in S'$*:*

$$\theta'(C_1, C_2) = \begin{cases} \mathsf{false} & \text{if } \simeq = \approx_{dbsf} \text{ and } C_1 = C_2 \\ \bigvee\{\, \theta(s,t) \mid s \in C_1 \wedge t \in C_2 \,\} & \text{otherwise} \end{cases}$$

- $L'([s]_\simeq) = L(s)$, *for all* $s \in S$.
- $s'_* = [s_*]_\simeq$.

The function that yields the naive $\simeq$-quotient for a feature expression $\varphi \in \mathbb{B}(\mathcal{F})$ of an FKS is denoted by $\overset{\varphi}{\simeq}$-$\min_{\mathtt{FKS}}$, for $\simeq \in \{\leftrightarrows_f, \approx_{dbsf}\}$. We use $\simeq$-$\min_{\mathtt{FKS}}$ as an abbreviation for $\overset{\mathsf{true}}{\simeq}$-$\min_{\mathtt{FKS}}$. An example of an FKS *fks* and its naive $\leftrightarrows_f$ -quotient *fks*$'$ are shown in Figure 5.1. We say that an FKS *fks* is *minimal modulo naive feature bisimulation* if and only if $\leftrightarrows_f$-$\min_{\mathtt{FKS}}(\mathit{fks}) = \mathit{fks}$. Similarly, we say it is *minimal modulo naive divergence-blind stuttering feature equivalence* if and only if $\approx_{dbsf}$-$\min_{\mathtt{FKS}}(\mathit{fks}) = \mathit{fks}$.

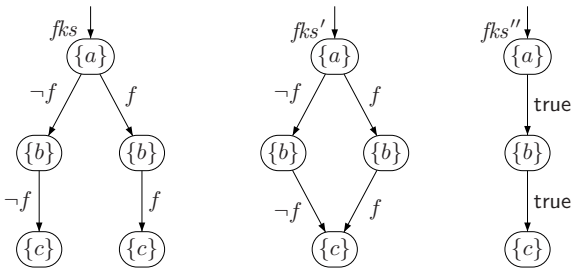


Figure 5.1: An FKS *fks*, its naive true-feature bisimulation quotient *fks*$'$, and its only coherent feature bisimulation quotient *fks*$''$.

We show that, for each feature expression ϕ, every FKS is feature bisimilar for ϕ to its naive feature bisimulation quotient for ϕ.

Theorem 5.1. *For each FKS fks, and for all* $\phi \in \mathbb{B}(\mathcal{F})$*, it holds that* $\overset{\phi}{\leftrightarrows}_f$-$\min_{\mathtt{FKS}}(\mathit{fks}) \ \overset{\phi}{\leftrightarrows}_f \ \mathit{fks}$.

Proof. Let $\mathit{fks} = (S, AP, \theta, L, s_*)$ be an FKS, let $\phi \in \mathbb{B}(\mathcal{F})$ be a feature expression, and let $\overset{\phi}{\leftrightarrows}_f$-$\min_{\mathtt{FKS}}(\mathit{fks}) = (S', AP, \theta', L', s'_*)$ be the naive feature bisimulation quotient for ϕ of *fks*. We define the relation R such that:

$$R = \{(s, [\varphi \wedge \phi]_{\sim_{\mathcal{P}}}, C) \in S \times \widehat{\mathbb{B}}(\mathcal{F}) \times S' \mid \exists t \in C \colon \mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t\} \cup$$
$$\{(C, [\varphi \wedge \phi]_{\sim_{\mathcal{P}}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S \mid \exists t \in C \colon \mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t\}$$

We have to show that R is a feature bisimulation relation on $\mathit{fks} \uplus \overset{\phi}{\leftrightarrows}_f$-$\min_{\mathtt{FKS}}(\mathit{fks})$ such that $(s_*, \hat{\phi}, s'_*) \in R$.

Since $\overset{\phi}{\leftrightarrows}_f$ is reflexive, we find that, for all $s \in S$, $(s, \widehat{\phi}, C) \in R$, where $C \in S'$ is such that $s \in C$. Using that $s_* \in s'_*$ by Definition 5.2, it immediately follows that $(s_*, \hat{\phi}, s'_*) \in R$. Hence, it remains to show that R is a feature bisimulation relation.

From the definition of R it immediately follows that it is symmetric, and hence it remains to show that R satisfies the conditions from Definition 4.2.

Consider a tuple $(s, [\varphi \wedge \phi]_{\sim_{\mathcal{P}}}, C) \in S \times \mathbb{B}(\mathcal{F}) \times S'$ such that $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$ and $t \in C$, for some $t \in S$. Let t be such.

We first show that $(L \uplus L')(s) = (L \uplus L')(C)$. Since $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$ we know that $L(s) = L(t)$, and since $t \in C$ we know that $L(t) = L'(C)$. Hence we can derive $(L \uplus L')(s) = L(s) = L(t) = L'(C) = (L \uplus L')(C)$.

Now we prove the transfer condition. Suppose that $s \xrightarrow{\psi} s'$, for some $\psi \in \mathbb{B}(\mathcal{F})$, and for some $s' \in S$. We first show that this transition can be mimicked by state t for all products satisfying $\varphi \wedge \psi$. Since $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$, there exists a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F})$ such that $\mathit{fks} \models s' \overset{\varphi'}{\leftrightarrows}_f t'$, for all $(t', \varphi') \in \mathcal{T}$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t',\varphi') \in \mathcal{T}} \theta(t, t') \wedge \varphi'.$$

Now we show that C can mimic all transitions from t, which we use to conclude that C can mimic the transition from s to s'. Using that $t \in C$, from Definition 5.2 it follows that $\theta(t, t') \Rightarrow_{\mathcal{P}} \theta'(C, C_{t'})$, with $t' \in C_{t'}$, for all $(t', \varphi') \in \mathcal{T}$. Since $t' \in C_{t'}$ and $\mathit{fks} \models s' \overset{\varphi'}{\leftrightarrows}_f t'$, by construction of R we have $(s', [\varphi' \wedge \phi]_{\sim_{\mathcal{P}}}, C_{t'}) \in R$, for all $(t', \varphi') \in \mathcal{T}$. We find that

$$\phi \wedge \bigvee_{(t',\varphi') \in \mathcal{T}} \theta(t, t') \wedge \varphi' \;\Rightarrow_{\mathcal{P}} \bigvee_{(t',\varphi') \in \mathcal{T}} \theta'(C, C_{t'}) \wedge \varphi' \wedge \phi,$$

and hence the transfer condition for this tuple is satisfied.

We prove the transfer conditions for the remaining tuples of R in two steps, using a case distinction.

1. First consider a tuple $(C, \hat{\phi}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \in C$. From Definition 5.2 it follows immediately that $(L \uplus L')(C) = (L \uplus L')(s)$.

   Suppose that $C \xrightarrow{\psi} C'$, for some $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. From Definition 5.2 it follows that $\psi = \bigvee\{\theta(u, u') \mid u \in C \wedge u' \in C'\}$. Pick a pair $(u, u') \in C \times C'$. Since both $s \in C$ and $u \in C$, by Definition 5.2 we know that $\mathit{fks} \models s \overset{\phi}{\leftrightarrows}_f u$. Hence we can find a set $\mathcal{S}_{(u,u')} \subseteq S \times \mathbb{B}(\mathcal{F})$ for each tuple $(u, u') \in C \times C'$ such that $\mathit{fks} \models s' \overset{\varphi'}{\leftrightarrows}_f u'$, for all $(s', \varphi') \in \mathcal{S}_{(u,u')}$, and such that

   $$\phi \wedge \theta(u, u') \;\Rightarrow_{\mathcal{P}} \bigvee_{(s',\varphi') \in \mathcal{S}_{(u,u')}} \theta(s, s') \wedge \varphi'.$$

   For each tuple $(u, u') \in C \times C'$ we have that $\mathit{fks} \models s' \overset{\varphi'}{\leftrightarrows}_f u'$, and hence by construction of R we have $(C', [\varphi' \wedge \phi]_{\sim_{\mathcal{P}}}, s') \in R$, for all $(s', \varphi') \in \mathcal{S}_{(u,u')}$. We find that

   $$\phi \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(u,u') \in C \times C'} \bigvee_{(s',\varphi') \in \mathcal{T}_{(u,u')}} \theta(s, s') \wedge \varphi' \wedge \phi,$$

   and hence the transfer condition for this tuple is satisfied.

2. Lastly consider a tuple $(C, [\varphi \wedge \phi]_{\sim_{\mathcal{P}}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \notin C$, and such that $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$ and $t \in C$, for some $t \in S$. Let t be such.

   We first show that $(L \uplus L')(C) = (L \uplus L')(s)$. Since $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$ we know that $L(s) = L(t)$, and since $t \in C$ we know that $L(t) = L'(C)$. Hence we can derive $(L \uplus L')(C) = L'(C) = L(t) = L(s) = (L \uplus L')(s)$.

Suppose that $C \xrightarrow{\psi} C'$, for some $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. By construction of R we have $(C, \hat{\phi}, t) \in R$. We have already shown that the transfer conditions are satisfied for such pairs in R, and hence we can find a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F})$ such that $(C', [\phi \wedge \varphi'_t]_{\sim_\mathcal{P}}, t') \in R$, for all $(t', \varphi'_t) \in \mathcal{T}$, and such that

$$\phi \wedge \psi \;\Rightarrow_\mathcal{P} \bigvee_{(t',\varphi'_t)\in\mathcal{T}} \theta(t,t') \wedge \varphi'_t \wedge \phi.$$

Now we show that s can mimic all transitions from t, which we use to conclude that s can mimic the transition from C to C'. Since $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$, we can find sets $\mathcal{S}_{t'} \subseteq S \times \mathbb{B}(\mathcal{F})$ for each $(t', \varphi'_t) \in \mathcal{T}$ such that, for all $(s', \varphi'_s) \in \mathcal{S}_{t'}$, we have $\mathit{fks} \models s' \overset{\varphi'_s}{\leftrightarrows}_f t'$ and such that

$$\varphi \wedge \theta(t,t') \;\Rightarrow_\mathcal{P} \bigvee_{(s',\varphi'_s)\in\mathcal{S}_{t'}} \theta(s,s') \wedge \varphi'_s.$$

Since $(C', [\phi \wedge \varphi'_t]_{\sim_\mathcal{P}}, t') \in R$, for all $(t', \varphi'_t) \in \mathcal{T}$, by definition of R this means that $\mathit{fks} \models t' \overset{\varphi'_t}{\leftrightarrows}_f u'$, for some $u' \in C'$. Since $\mathit{fks} \models s' \overset{\varphi'_s}{\leftrightarrows}_f t'$ for all $(s', \varphi'_s) \in \mathcal{S}_{t'}$, we derive $\mathit{fks} \models s' \overset{\varphi'_s \wedge \varphi'_t}{\leftrightarrows}_f u'$ using productwise transitivity. By construction of R we have $(C', [\phi \wedge \varphi'_s \wedge \varphi'_t]_{\sim_\mathcal{P}}, s') \in R$. We find that

$$\varphi \wedge \bigvee_{(t',\varphi'_t)\in\mathcal{T}} \theta(t,t') \wedge \varphi'_t \wedge \phi \;\Rightarrow_\mathcal{P} \bigvee_{(t',\varphi'_t)\in\mathcal{T}} \bigvee_{(s',\varphi'_s)\in\mathcal{S}_{t'}} \theta(s,s') \wedge \varphi'_s \wedge \varphi'_t \wedge \phi,$$

and hence that

$$\varphi \wedge \phi \wedge \psi \;\Rightarrow_\mathcal{P} \bigvee_{(t',\varphi'_t)\in\mathcal{T}} \bigvee_{(s',\varphi'_s)\in\mathcal{S}_{t'}} \theta(s,s') \wedge \varphi'_s \wedge \varphi'_t \wedge \phi,$$

from which we conclude that the transfer condition for this pair is also satisfied.

□

Similarly we show that, for each feature expression ϕ, every FKS is divergence-blind stuttering feature equivalent for ϕ to its naive divergence-blind stuttering feature quotient for ϕ. As this theorem is similar to Theorem 5.1, the proof is omitted here. The interested reader is referred to Appendix C.

Theorem 5.2. *For each FKS fks, and for all $\phi \in \mathbb{B}(\mathcal{F})$, it holds that $\overset{\phi}{\approx}_{dbsf}\text{-min}_{\texttt{FKS}}(\mathit{fks}) \overset{\phi}{\approx}_{dbsf} \mathit{fks}$.*

The approach sketched above is naive since it requires that two states are equivalent for every product satisfying the feature expression φ in order to be mapped to the same state in the quotient. However, since we are considering FKS with an initial state, not all states are reachable for all products, meaning some states do not specify any behavior for some products. Hence it suffices to require that two states are equivalent for all the products *they have in common* in order to be mapped to the same state in the quotient. To specify this we use the reachability function ϱ, as specified in Definition 3.5.

Definition 5.3. *Let fks* $= (S, AP, \theta, L, s_*)$ *be an FKS, and let* ϱ *denote its reachability function. We say two states* $s, t \in S$ *are coherent feature bisimilar or coherent divergence-blind stuttering feature equivalent, denoted by fks* $\models s \leftrightarrows_{cf} t$ *and fks* $\models s \approx_{cdbsf} t$*, if and only if fks* $\models s \overset{\varrho(s)\wedge\varrho(t)}{\simeq} t$*, for* $\simeq \in \{\leftrightarrows_f, \approx_{dbsf}\}$*, respectively.*

The relations $\leftrightarrows_{cf}$ and $\approx_{cdbsf}$ are not transitive and therefore different 'equivalence classes' induced by these relations (maximal cliques in the graph induced by $\leftrightarrows_{cf}$ or $\approx_{cdbsf}$) may have a non-empty intersection. Because the relations are not transitive in general, unique quotients may not exist. This phenomenon is illustrated in Figure 5.2. In this example the state with label $\{a\}$ and the state with label $\{c\}$ both belong to equivalence classes consisting of one state. Furthermore we have three equivalence classes each consisting of two states with label $\{b\}$, where the inner states with label $\{b\}$ both belong to two of those classes.

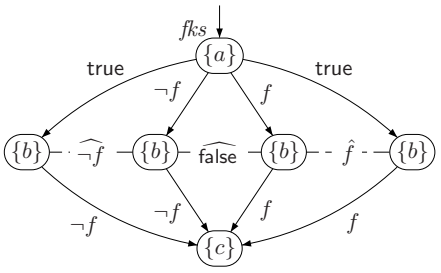


Figure 5.2: An FKS with the maximal coherent feature bisimulation relation. Reflexive parts of the relation are omitted.

We proceed by giving the definition of *a* coherent quotient of an FKS. All quotients adhering to this definition are included in *the set* of coherent quotients of this FKS. The state space of a coherent quotient is constructed by assigning each state to exactly one of its 'equivalence classes', in such a way that the remaining number of non-empty classes is minimal. Since it may be possible to do this in multiple ways, this part of the definition is non-deterministic. The transition-constraint function of a coherent quotient is strengthened with the reachability information of the original FKS. This is necessary since we assumed that a state can only define behavior for the products included in its reachability set, but it is possible that the transition-constraint function of the original FKS does not respect this assumption.

Definition 5.4. *Let fks* $= (S, AP, \theta, L, s_*)$ *be an FKS with reachability function* ϱ*, and let* $\sim \in \{\leftrightarrows_{cf}, \approx_{cdbsf}\}$*. A* $\sim$*-quotient of fks is an FKS fks*$' = (S', AP, \theta', L', s'_*)$*, such that*

- $S' \subseteq \bigcup_{C \in \langle S \rangle_\sim} 2^C$*, such that*
  1. $\bigcup S' = S$*;*
  2. $C_1 \cap C_2 = \emptyset$*, for all distinct* $C_1, C_2 \in S'$*;*
  3. $|S'|$ *is minimal.*
- $\theta' \colon S' \times S' \to \mathbb{B}(\mathcal{F})$ *is constructed such that, for all* $C_1, C_2 \in S'$*:*

$$\theta'(C_1, C_2) = \begin{cases} \mathsf{false} & \text{if } \sim = \approx_{cdbsf} \text{ and } C_1 = C_2 \\ \bigvee\{\, \theta(s,t) \wedge \varrho(s) \mid s \in C_1 \wedge t \in C_2 \,\} & \text{otherwise} \end{cases}$$

- $L'([s]_{\simeq_{S'}}) = L(s)$, *for all* $s \in S$.
- $s'_* = [s_*]_{\simeq_{S'}}$.

In the construction of the state space of the coherent quotient we demand that $|S'|$ is minimal. Figure 5.2 illustrates why this is necessary. The FKS in this figure would have five coherent quotients without this requirement: three of which would have a state space of size five, one with a state space of size six, and the last one would have a state space of size four. By adding the requirement that $|S'|$ is minimal, the only coherent quotient of this FKS is the quotient with a state space of size four.

Note that $fks \models s \simeq t$ implies $fks \models s \sim t$, for $(\simeq, \sim) \in \{(\leftrightarrows_f, \leftrightarrows_{cf}), (\approx_{dbsf}, \approx_{cdbsf})\}$, by Definition 5.3. It follows that each FKS that is minimal modulo $\sim$ is also minimal modulo $\simeq$. Hence the naive quotient of an FKS is at least as large as its coherent quotients.

The function that yields the set of $\sim$-quotients of an FKS is denoted by $\sim$-$\min_{\mathtt{FKS}}$, for $\sim \in \{\leftrightarrows_{cf}, \approx_{cdbsf}\}$. An example of an FKS and its (unique) $\leftrightarrows_{cf}$-quotient are shown in the left and right parts of Figure 5.1, respectively. We say that an FKS *fks* is *minimal modulo coherent feature bisimulation* if and only if $\leftrightarrows_{cf}$-$\min_{\mathtt{FKS}}(fks) = \{fks\}$. Similarly, we say it is *minimal modulo coherent divergence-blind stuttering feature equivalence* if and only if $\approx_{cdbsf}$-$\min_{\mathtt{FKS}}(fks) = \{fks\}$.

We show that every FKS is feature bisimilar to its coherent feature bisimulation quotients.

Theorem 5.3. *For each FKS fks and for all* $fks' \in \leftrightarrows_{cf}$-$\min_{\mathtt{FKS}}(fks)$, *it holds that* $fks' \leftrightarrows_f fks$.

Proof. The proof is similar to that of Theorem 5.1 for the naive quotient. The full proof can be found in Appendix C. □

Similarly we show that every FKS is divergent-blind stuttering feature equivalent to its coherent divergent-blind stuttering feature quotients. As with Theorems 5.2 and 5.3, the proof for this theorem can be found in Appendix C.

Theorem 5.4. *For each FKS fks and for all* $fks' \in \approx_{cdbsf}$-$\min_{\mathtt{FKS}}(fks)$, *it holds that* $fks' \approx_{dbsf} fks$.

We continue with the quotients for event-based models.

5.2 Event-based models

The theorems in this section are similar to those of Section 5.1. Hence, the proofs are omitted. The interested reader can find them in Appendix C.

We first define the bisimulation quotient for LTS:

Definition 5.5. *Let* $lts = (S, \mathcal{A}, \rightarrow, s_*)$ *be an LTS, and let* $\simeq \in \{\leftrightarrows, \leftrightarrows_b\}$. *The* $\simeq$*-quotient of lts is the LTS* $lts' = (S', \mathcal{A}, \rightarrow', s'_*)$, *such that*

- $S' = [S]_{\simeq}$.

- *If* $\simeq = \leftrightarrows$, *we have*

$$\rightarrow' = \{\, (C_1, \alpha, C_2) \in S' \times \mathcal{A}_\tau \times S' \mid \exists (s,t) \in C_1 \times C_2 \colon (s, \alpha, t) \in \rightarrow \,\}.$$

  Otherwise, if $\simeq = \leftrightarrows_b$, *we have*

$$\rightarrow' = \{\, (C_1, \alpha, C_2) \in S' \times \mathcal{A}_\tau \times S' \mid (C_1 \neq C_2 \vee \alpha \neq \tau) \wedge \exists (s,t) \in C_1 \times C_2 \colon (s, \alpha, t) \in \rightarrow \,\}.$$

- $s'_* = [s_*]_\simeq$.

The function that yields the $\simeq$-quotient of an LTS is denoted by $\simeq$-$\min_{\mathtt{LTS}}$, for $\simeq \in \{\leftrightarrows, \leftrightarrows_{\mathrm{b}}\}$. We say that an LTS *lts* is *minimal modulo bisimulation* if and only if $\leftrightarrows$-$\min_{\mathtt{LTS}}(lts) = lts$. Similarly, we say it is *minimal modulo branching bisimulation* if and only if $\leftrightarrows_{\mathrm{b}}$-$\min_{\mathtt{LTS}}(lts) = lts$.

We will extend this definition to FTS in the same ways as we extended the quotients for KS to quotients for FKS. As with FKS, we first define the naive quotients.

Definition 5.6. *Let* $fts = (S, \mathcal{A}, \theta, s_*)$ *be an FTS, let* $\varphi \in \mathbb{B}(\mathcal{F})$ *be a feature expression, and let* $\simeq \in \{\leftrightarrows_f, \leftrightarrows_{bf}\}$. *The naive* $\simeq$*-quotient of fts for* φ *is the FTS* $fts' = (S', \mathcal{A}, \theta', s'_*)$, *such that*

- $S' = [S]_{\overset{\varphi}{\simeq}}$.
- $\theta' \colon S' \times \mathcal{A}_\tau \times S' \to \mathbb{B}(\mathcal{F})$ *is constructed such that, for all* $C_1, C_2 \in S'$ *and* $\alpha \in \mathcal{A}_\tau$*:*

$$\theta'(C_1, \alpha, C_2) = \begin{cases} \mathsf{false} & \text{if } \simeq = \leftrightarrows_{bf} \text{ and } C_1 = C_2 \text{ and } \alpha = \tau \\ \bigvee\{\, \theta(s, \alpha, t) \mid s \in C_1 \wedge t \in C_2 \,\} & \text{otherwise} \end{cases}$$

- $s'_* = [s_*]_\simeq$.

The function that yields the naive $\simeq$-quotient for a feature expression $\varphi \in \mathbb{B}(\mathcal{F})$ of an FTS is denoted by $\overset{\varphi}{\simeq}$-$\min_{\mathtt{FTS}}$, for $\simeq \in \{\leftrightarrows_f, \leftrightarrows_{bf}\}$. We say that an FTS *fts* is *minimal modulo naive feature bisimulation* if and only if $\leftrightarrows_f$-$\min_{\mathtt{FTS}}(fts) = fts$. Similarly, we say it is *minimal modulo naive branching feature bisimulation* if and only if $\leftrightarrows_{bf}$-$\min_{\mathtt{FTS}}(fts) = fts$.

We show that, for each feature expression ϕ, every FTS is feature bisimilar for ϕ to its naive feature bisimulation quotient for ϕ.

Theorem 5.5. *For each FTS fts, and for all* $\phi \in \mathbb{B}(\mathcal{F})$, *it holds that* $\overset{\phi}{\leftrightarrows}_f$-$\min_{\mathtt{FTS}}(fts)$ $\overset{\phi}{\leftrightarrows}_f$ *fts.*

Similarly we show that, for each feature expression ϕ, every FTS is branching feature bisimilar for ϕ to its naive branching feature bisimulation quotient for ϕ.

Theorem 5.6. *For each FTS fts, and for all* $\phi \in \mathbb{B}(\mathcal{F})$, *it holds that* $\overset{\phi}{\leftrightarrows}_{bf}$-$\min_{\mathtt{FTS}}(fts)$ $\overset{\phi}{\leftrightarrows}_{bf}$ *fts.*

Next we define when two states in an FTS are *coherent feature bisimilar* or *coherent branching feature bisimilar*.

Definition 5.7. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, and let ϱ denote its reachability function. We say two states $s, t \in S$ are coherent feature bisimilar or coherent branching feature bisimilar, denoted by $fts \models s \leftrightarrows_{cf} t$ and $fts \models s \leftrightarrows_{cbf} t$, if and only if $fts \models s \overset{\varrho(s)\wedge\varrho(t)}{\simeq} t$, for $\simeq \in \{\leftrightarrows_f, \leftrightarrows_{bf}\}$, respectively.*

As with FKS, these relations are not transitive and therefore different 'equivalence classes' induced by these relations (maximal cliques in the graph induced by $\leftrightarrows_{cf}$ or $\leftrightarrows_{cbf}$) may have a non-empty intersection. Hence, unique coherent quotients do not exist. We proceed by giving the definition of a coherent quotient of an FTS. All quotients adhering to this definition are included in the set of coherent quotients of this FTS.

Definition 5.8. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS with reachability function ϱ, and let $\sim \in \{\leftrightarrows_{cf}, \leftrightarrows_{cbf}\}$. A $\sim$-quotient of fts is an FTS $fts' = (S', \mathcal{A}, \theta', s'_*)$, such that*

- *$S' \subseteq \bigcup_{C \in \langle S \rangle_\sim} 2^C$, such that*
  1. *$\bigcup S' = S$;*
  2. *$C_1 \cap C_2 = \emptyset$, for all distinct $C_1, C_2 \in S'$;*
  3. *$|S'|$ is minimal.*
- *$\theta' \colon S' \times \mathcal{A}_\tau \times S' \to \mathbb{B}(\mathcal{F})$ is constructed such that, for all $C_1, C_2 \in S'$ and $\alpha \in \mathcal{A}_\tau$:*

$$\theta'(C_1, C_2) = \begin{cases} \mathsf{false} & \text{if } \sim = \leftrightarrows_{bf} \text{ and } C_1 = C_2 \text{ and } \alpha = \tau \\ \bigvee\{\, \theta(s, \alpha, t) \wedge \varrho(s) \mid s \in C_1 \wedge t \in C_2 \,\} & \text{otherwise} \end{cases}$$

- *$s'_* = [s_*]_{\simeq_{S'}}$.*

The function that yields the set of $\sim$-quotients of an FTS is denoted by $\sim\text{-min}_{\mathtt{FTS}}$, for $\sim \in \{\leftrightarrows_{cf}, \leftrightarrows_{cbf}\}$. We say that an FTS *fts* is *minimal modulo coherent feature bisimulation* if and only if $\leftrightarrows_{cf}\text{-min}_{\mathtt{FTS}}(fts) = \{fts\}$. Similarly, we say it is *minimal modulo coherent branching feature bisimulation* if and only if $\leftrightarrows_{cbf}\text{-min}_{\mathtt{FTS}}(fts) = \{fts\}$.

We show that every FTS is feature bisimilar to its coherent feature bisimulation quotients.

Theorem 5.7. *For each FTS fts and for all $fts' \in \leftrightarrows_{cf}\text{-min}_{\mathtt{FTS}}(fts)$ it holds that $fts' \leftrightarrows_f fts$.*

Similarly, we show that every FTS is branching feature bisimilar to its coherent branching feature bisimulation quotients.

Theorem 5.8. *For each FTS fts and for all $fts' \in \leftrightarrows_{cbf}\text{-min}_{\mathtt{FTS}}(fts)$ it holds that $fts' \leftrightarrows_{bf} fts$.*

Note that the coherent $\sim$-quotients of an FTS *fts* are not always minimal in the number of states modulo $\simeq$, for $(\simeq, \sim) \in \{(\leftrightarrows_f, \leftrightarrows_{cf}), (\leftrightarrows_{bf}, \leftrightarrows_{cbf})\}$. An example is shown in Figure 5.3. The FTS fts' is branching feature bisimilar to the FTS *fts*, and has a state space of size 2. However, the FTS fts'' is one of the coherent branching feature bisimulation quotients of *fts*, and has a state space of size 3.

This example suggests that it is in general not possible to construct a minimal FTS modulo (branching) feature bisimulation without allowing to split states. However, allowing to split states can easily lead to a growth in the number of transitions, even when keeping the number of states minimal, as shown in Figure 5.4.

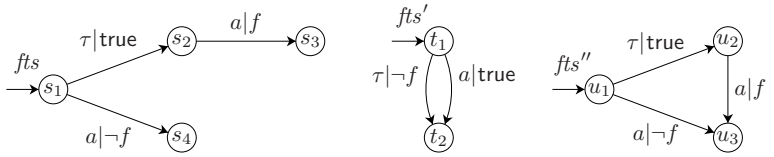


Figure 5.3: An FTS *fts*, its minimal representation modulo branching feature bisimulation *fts′*, and one of its coherent branching feature bisimulation quotients *fts″*.

Such a result is unwanted as it 'breaks up' the family-level behavior of the system. Hence, we leave the challenge of defining a quotient that is truly minimal (in both states and transitions) modulo (branching) feature bisimulation for future work, as well as establishing whether such a quotient is actually unique. The above discussion is also applicable to FKS, for relations $(\simeq, \sim) \in \{(\leftrightarrows_f, \leftrightarrows_{cf}), (\approx_{dbsf}, \approx_{cdbsf})\}$.

We now give an overview of the the relations between the different quotients defined on the product-family level. In general, let $\mathcal{S}$ be an FKS or FTS, and let the tuple $(\simeq, \sim)$ be in $\{(\leftrightarrows_f, \leftrightarrows_{cf}), (\approx_{dbsf}, \approx_{cdbsf})\}$ or $\{(\leftrightarrows_f, \leftrightarrows_{cf}), (\leftrightarrows_{bf}, \leftrightarrows_{cbf})\}$, respectively. Let $|\mathcal{S}|$ denote the number of states of the transition systems $\mathcal{S}$. For a concrete example, we pick $\mathcal{S}$ as the FTS *fts*, and $(\simeq, \sim)$ as $(\leftrightarrows_f, \leftrightarrows_{cf})$.

We let $[fts]_{\leftrightarrows_f} = \{fts' \mid fts \leftrightarrows_f fts'\}$ denote the set of all FTS that are *feature bisimilar* to *fts*. We have the following properties regarding the coherent quotients.

1. The FTS *fts* has a set of *coherent feature bisimulation quotients* $\leftrightarrows_{cf}\text{-min}_{\mathtt{FTS}}(fts)$. Each of these quotients has the same number of states, by Definition 5.8.

2. We have $\leftrightarrows_{cf}\text{-min}_{\mathtt{FTS}}(fts) \subseteq [fts]_{\leftrightarrows_f}$ by Theorem 5.7.

3. An FTS *fts′* that is minimal modulo *coherent* feature bisimulation does not have to be minimal modulo feature bisimulation. That is, there may be an $fts'' \in [fts]_{\leftrightarrows_f}$ such that $|fts''| < |fts'|$. An example is shown in Figure 5.3.

4. Conversely, each FTS that is minimal modulo feature bisimulation is also minimal modulo *coherent* feature bisimulation. This follows directly from point 2.

Lastly, we have the following properties regarding the naive quotients.

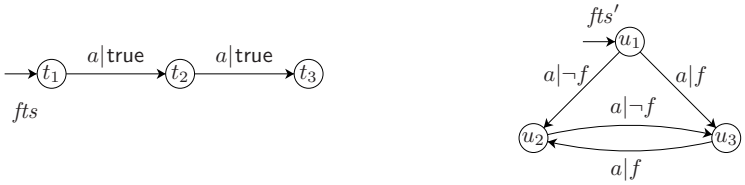


Figure 5.4: Two feature bisimilar FTS.

1. The FTS *fts* has exactly one *naive feature bisimulation quotient* $\overset{\mathsf{true}}{\longleftrightarrow}_f\text{-min}_{\mathtt{FTS}}(fts)$.
2. We have $\overset{\mathsf{true}}{\longleftrightarrow}_f\text{-min}_{\mathtt{FTS}}(fts) \in [fts]_{\underline{\leftrightarrow}_f}$ by Theorem 5.5.
3. An FTS fts' that is minimal modulo *naive* feature bisimulation does not have to be minimal modulo *coherent* feature bisimulation. That is, it may be the case that $|\ \underline{\leftrightarrow}_{cf}\text{-min}_{\mathtt{FTS}}(fts')| < |fts'|$. An example is shown in Figure 5.1.
4. Conversely, each FTS that is minimal modulo *coherent* feature bisimulation is also minimal modulo *naive* feature bisimulation. This follows directly the definition of coherent feature bisimulation (Definition 5.7.

We conclude this chapter with a section on related work.

5.3 Related work

In [11], Cordy et al. propose simulation-equivalence quotients for their Featured Transition Systems. Similar to our work, their first proposal is a quotient that merges states that are simulation equivalent for all products. Hence, this quotient is equivalent to our naive true-quotients. Experiments performed by Cordy et al. showed that this quotient does not result in useful reduction in practice, and hence they defined a second quotient that takes the reachability function into account. For this purpose they defined that 1) a state s trivially simulates a state t for all products that cannot reach t, and that 2) a state s cannot simulate a state t for products that can reach t but not s.

Using this definition a new equivalence relation is defined, where two states are reachability-aware simulation equivalent iff they can simulate each other for all products. Note that this implies that the reachability sets of two states must be equal for those two states to be reachability-aware simulation equivalent. Hence, this second equivalence is much finer than our definition of coherent equivalence, and is therefore much less powerful. It is, however, more powerful than the naive quotient.

Lastly a quotient was proposed based on the featured simulation *preorder*, again taking reachability into account. This quotients is, like our coherent quotient, not unique, which makes it more difficult to define than the previous two quotients. Moreover, since this quotient is not based on an equivalence, behavior may be added during the state space reduction process. Hence, validity of proporties is not always preserved by this quotient, which may be problematic.

Chapter 6

An algorithm for coherent branching feature bisimulation

In this chapter we present an algorithm [7] to calculate a coherent branching feature bisimulation quotient for a given FTS. This algorithm is based on the algorithm for branching bisimilarity as described by Groote and Vaandrager [18]. The coherent branching feature bisimulation quotient does not allow to 'split' a state from the original system over its features and map different parts of the state to different states in the reduced FTS. Hence, as argued in the previous chapter, the 'minimal' representations of FTS calculated by the algorithm may actually not be minimal modulo branching feature bisimulation. However, as we will see, even for this restricted class of quotients the addition of the feature dimension to the system introduces multiple challenges for the adaptation of the algorithm, making this a far from trivial problem.

In Section 6.1 we discuss the complexity of coherent feature bisimulation reduction in general, and we prove that this problem is NP-hard. In Section 6.2 we present our adaptation of the Groote-Vaandrager algorithm. Lastly, Section 6.3 contains implementation details of the algorithm, including detailed pseudo-code.

6.1 Complexity of coherent feature bisimulation reduction

The traditional algorithm for minimization modulo branching bisimilarity maintains a partition of the states of an LTS. It keeps refining this partition until it satisfies the transfer conditions of branching bisimilarity, which means a branching bisimulation relation has been found. This approach uses the transitivity of the branching bisimulation relation: if state s_1 is branching bisimilar to state s_2, but not to state s_3, then we also know that state s_2 is not branching bisimilar to s_3. Hence, it is always possible to split a block of the partition in two strict subsets, which means the result is again a partition.

With the addition of the feature dimension, our goal is to calculate a coherent branching feature bisimulation quotient, which means we are relating different states for different sets of products. As a result, it is not possible to maintain a partition of the state space as done by the original algorithm, since a violation of the transfer condition does not necessarily result in a division of the state space into two non-overlapping subsets. This can also be observed from

Definition 5.8 of the coherent branching feature bisimulation quotient, where we used $\langle S \rangle \underline{\leftrightarrows}_{cbf}$ instead of $[S]\underline{\leftrightarrows}_{cbf}$ to determine its state space. As a result the complexity of the minimization process increases significantly, which we show by proving that the problem of finding a coherent (branching) feature bisimulation quotient is NP-hard by reducing the chromatic number problem to it: given a graph, what is the minimum number of colors to color the nodes such that adjacent nodes have different colors?

Consider undirected connected graph $\mathcal{G} = (V, E)$ with V the set of nodes with $|V| \geq 2$, and $E \subseteq V \times V$ the set of edges. Let $\mathcal{F} = \{ f_v \mid v \in V \}$ and let $\mathcal{P} = \{ P_v \mid v \in V \}$ such that $P_v \models f_u \Leftrightarrow u = v$. Let the FTS $fts_G = (S_G, \mathcal{A}, \theta_G, s_*)$ of $\mathcal{G}$ be such that

- $S_G = \{s_*, s_\perp\} \cup \{ s_v \mid v \in V \}$, for distinguished states s_* and $s_\perp$.
- $\mathcal{A} = \{a\}$.
- $\theta_G \colon S_G \times \mathcal{A}_\tau \times S_G \to \mathbb{B}(\mathcal{F})$ is constructed such that, for all $s, t \in S_G$ and $\alpha \in \mathcal{A}_\tau$:

$$\theta_G(s, \alpha, t) = \begin{cases} \bigvee_{u \in V} \{ f_u \mid (u, v) \in \mathcal{G} \} \vee f_v & \text{if } s = s_* \text{ and } \exists v \in V : t = s_v \text{ and } \alpha = a \\ f_v & \text{if } \exists v \in V : s = s_v \text{ and } t = s_\perp \text{ and } \alpha = a \\ \mathsf{false} & \text{otherwise} \end{cases}$$

We will show that the number of states of a coherent feature bisimulation quotient of fts_G is equal to the chromatic number of $\mathcal{G}$ plus 2. Since fts_G does not contain any silent transitions the obtained result is also applicable to the coherent branching feature bisimulation quotient.

We first prove a lemma stating that two states $s, t \in S_G$ are coherent feature bisimilar if and only if there exist vertices $u, v \in V$ such that $s = s_u$ and $t = s_v$, and such that u and v are not neighbors in $\mathcal{G}$.

Lemma 6.1. *Let the graph $\mathcal{G} = (V, E)$ and the FTS $fts_G = (S_G, \mathcal{A}_G, \theta_G, s_{*G})$ be as given above. For distinct $s, t \in S_G$, we have that*

$$fts_G \models s \underline{\leftrightarrows}_{cf} t \Leftrightarrow (\exists u, v \in V : s = s_u \wedge t = s_v \wedge (u, v) \notin E).$$

Proof. The proof for this lemma is included in Appendix D. □

We now prove the main theorem.

Theorem 6.1. *Let fts'_G be a coherent feature bisimulation quotient of the FTS fts_G given above. Then the number of states in fts'_G is equal to the chromatic number of $\mathcal{G}$ plus 2.*

Proof. Let $\mathcal{G} = (V, E)$ be as given above. Let $fts'_G = (S'_G, \mathcal{A}_G, \theta'_G, s'_{*G})$ be a coherent feature bisimulation quotient of the FTS $fts_G = (S_G, \mathcal{A}_G, \theta_G, s_{*G})$ given above.

Consider S'_G as a set of colors. The state space S'_G is a partition of S_G, and hence it induces a coloring $\gamma \colon V \to S'_G$ for $\mathcal{G}$, where $\gamma(u) = \gamma(v)$ if and only if $[s_u]_{\simeq_{S'_G}} = [s_v]_{\simeq_{S'_G}}$, for all $u, v \in V$. If $[s_u]_{\simeq_{S'_G}} = [s_v]_{\simeq_{S'_G}}$, then $fts_G \models s_u \underline{\leftrightarrows}_{cf} s_v$, which implies that $(u, v) \notin E$ by Lemma 6.1. Hence, γ is a valid coloring of $\mathcal{G}$. It remains to show that γ is a minimal coloring. That is, we have to show that $|\{\gamma(v) \mid v \in V\}|$ is minimal. From Lemma 6.1 it follows that $[s_*]_{\simeq_{S'_G}} = \{s_*\}$, and that $[s_\perp]_{\simeq_{S'_G}} = \{s_\perp\}$, from which we derive that $|\{\gamma(v) \mid v \in V\}| = |S'_G| - 2$.

Assume we can find a set of colors Γ such that $|\Gamma| < |\{\gamma(v) \mid v \in V\}|$, and a valid coloring $\gamma' \colon V \to \Gamma$ of $\mathcal{G}$ using all colors. We derive a contradiction by showing that fts'_G is not a coherent feature bisimulation quotient of fts_G.

For every pair of vertices $u, v \in V$ we have that $\gamma'(u) = \gamma'(v)$ implies $(u, v) \notin E$. Since γ' is a valid coloring of $\mathcal{G}$ by assumption, from Lemma 6.1 it follows that $fts_G \models s_u \leftrightarrows_{cf} s_v$. Hence, the set $S''_G = \bigcup_{v \in V} \{u \in V \mid \gamma'(u) = \gamma'(v)\} \cup \{s_*\} \cup \{s_\perp\}$ is a partition of S_G such that $[s]_{\simeq_{S''_G}} = [t]_{\simeq_{S''_G}}$ implies $fts_G \models s \leftrightarrows_{cf} t$, for all $s, t \in S_G$. We find that

$$|S''_G| = |\Gamma| + 2 < |\{\gamma(v) \mid v \in V\}| + 2 = |S'_G| - 2 + 2 = |S'_G|,$$

from which it follows that fts'_G is not a coherent feature bisimulation quotient of fts_G.

From the contradiction obtained we derive that $|\{\gamma(v) \mid v \in V\}|$ is the chromatic number of $\mathcal{G}$, from which we conclude that the number of states in fts'_G is indeed equal to the chromatic number of $\mathcal{G}$ plus 2.

□

From the theorem we obtain the following result.

Corollary 6.1. *Constructing a minimal coherent (branching) feature bisimilar FTS is NP-hard.* □

In the next section we present the algorithm for coherent branching feature bisimulation minimization for FTS.

6.2 Coherent branching feature bisimulation minimization

As stated in the previous section, due to the nature of coherent branching feature bisimulation it is not possible to maintain a strict partition of the state space while calculating its quotient. As a solution, we introduce the concept of a semi-partition and transform the branching bisimilarity minimization algorithm based on this concept.

A collection $\mathcal{B} = \{ B_i \mid i \in I \}$ of non-empty subsets of a set S is called a *semi-partition* of S if (i) $\bigcup_{i \in I} B_i = S$, and (ii) for $j \neq i : B_j \setminus B_i \neq \varnothing$. Thus, $\mathcal{B}$ covers S and no B_j is strictly contained in a B_i. Also, for a semi-partition its elements are referred to as *blocks*. We say that a semi-partition $\mathcal{B}'$ is a refinement of a semi-partition $\mathcal{B}$ if every block of $\mathcal{B}'$ is a subset of a block of $\mathcal{B}$. Likewise, we say that $\mathcal{B}$ is coarser than $\mathcal{B}'$. A semi-partition $\mathcal{B}$ of S induces a relation $\sim_{\mathcal{B}}$ on S (not necessarily an equivalence relation), where two elements of S are related iff they are included in the same block of $\mathcal{B}$.

Given an FTS $fts = (S, \mathcal{A}, \theta, s_*)$ with reachability function ϱ, we first do some pre-processing. We eliminate unreachable states and strengthen the transition constraint function with the reachability condition for its source state:

$$S := \{ s \in S \mid \varrho(s) \not\sim_{\mathcal{P}} \mathsf{false} \} \quad \text{and} \quad \theta(s, \alpha, s') := \theta(s, \alpha, s') \wedge \varrho(s)$$

Note that this pre-processing step is quite expensive, as determining reachability in FTS is NP-complete [9]. Hence, calculating the entire reachability function is NP-hard. However, the reachability function has a crucial role in the definition of coherent branching feature bisimilarity, and therefore calculating it is unavoidable.

We define the set $\mathcal{A}_f$ of so-called featured labels by $\mathcal{A}_f = \{\ (\alpha,\psi) \mid \exists s,t \in S\colon s \xrightarrow{\alpha|\psi} t\ \}$. For a semi-partition $\mathcal{B}$ of S, $B, B' \in \mathcal{B}$ and featured label $(\alpha,\psi) \in \mathcal{A}_f$ we let

$$\begin{aligned} &\textit{non-neg}_{(\alpha,\psi)}(B,B') \ = \\ &\qquad \{\ s \in B \mid \forall P \in \mathcal{P},\, P \models \varrho(s) \wedge \psi\colon \exists n\, \exists s_0,\ldots,s_n \in B\, \exists s' \in B'\, \exists \psi_1,\ldots,\psi_n,\psi'\colon \\ &\qquad\qquad s_0 = s \wedge (\forall i, 1 \le i \le n\colon s_{i-1} \xrightarrow{\tau|\psi_i} s_i \wedge P \models \psi_i) \wedge s_n \xrightarrow{(\alpha|\psi')} s' \wedge P \models \psi'\ \}, \end{aligned}$$

and define its subset $\textit{pos}_{(\alpha,\psi)}(B,B')$ to include all states $s \in \textit{non-neg}_{(\alpha,\psi)}(B,B')$ that are reachable for all products satisfying ψ, and that reach block B' using a real step. That is, if $\alpha = \tau$ and $s \in B \cap B'$, then s is only in $\textit{pos}_{(\alpha,\psi)}(B,B')$ if it can reach B' using a real τ-transition.

$$\begin{aligned} \textit{pos}_{(\alpha,\psi)}(B,B') \ = \ \{\ s \in B \mid \psi \Rightarrow_{\mathcal{P}} \varrho(s) \wedge \\ \forall P \in \mathcal{P},\, P \models \psi\colon \exists n\, \exists s_0,\ldots,s_n \in B\, \exists s' \in B'\, \exists \psi_1,\ldots,\psi_n,\psi'\colon \\ s_0 = s \wedge (\forall i, 1 \le i \le n\colon s_{i-1} \xrightarrow{\tau|\psi_i} s_i \wedge P \models \psi_i) \wedge s_n \xrightarrow{\alpha|\psi'} s' \wedge P \models \psi'\ \}. \end{aligned}$$

Moreover, we define $\textit{neg}_{(\alpha,\psi)}(B,B') = B \setminus \textit{non-neg}_{(\alpha,\psi)}(B,B')$. We know for sure that two states s and t of a block B are behaviorally different, if $s \in \textit{pos}_{(\alpha,\psi)}(B,B')$ and $t \in \textit{neg}_{(\alpha,\psi)}(B,B')$. Therefore, we say that B' is a *splitter* of B with respect to (α,ψ) if $B \neq B'$ or $\alpha \neq \tau$, and $\textit{pos}_{(\alpha,\psi)}(B,B'), \textit{neg}_{(\alpha,\psi)}(B,B') \neq \varnothing$. If $\mathcal{B}$ is a semi-partition of S and B' is a splitter of B with respect to (α,ψ), then the semi-partition $\mathcal{B}'$ is obtained from $\mathcal{B}$ by replacing block B by $B_1 = \textit{non-neg}_{(\alpha,\psi)}(B,B')$ and $B_2 = B \setminus \textit{pos}_{(\alpha,\psi)}(B,B')$. However, in the case that B_1 or B_2 is a subset of another block in the partition (apart from B), it is not added to ensure that $\mathcal{B}'$ is a semi-partition.

The minimization algorithm starts from the trivial semi-partition $\{S\}$, and keeps refining the semi-partition until no splitters are left.

$\mathcal{B} := \{S\}$;
while a splitter B' for a block B with respect to a featured label (α,ψ) exists **do**
  $\mathcal{B} := \mathcal{B} \setminus \{B\}$;
  if $\textit{non-neg}_{(\alpha,\psi)}(B,B') \subseteq B''$ for no $B'' \in \mathcal{B}$
    then $\mathcal{B} := \mathcal{B} \cup \{\textit{non-neg}_{(\alpha,\psi)}(B,B')\}$;
  end;
  if $B \setminus \textit{pos}_{(\alpha,\psi)}(B,B') \subseteq B''$ for no $B'' \in \mathcal{B}$
    then $\mathcal{B} := \mathcal{B} \cup \{B \setminus \textit{pos}_{(\alpha,\psi)}(B,B')\}$;
  end;

It is easy to see that the algorithm terminates. The resulting semi-partition $\mathcal{B}'$ after each iteration is a strict refinement of the semi-partition $\mathcal{B}$ before this iteration. That is, $\mathcal{B}'$ is a refinement of $\mathcal{B}$ and $\mathcal{B}' \neq \mathcal{B}$. It is not possible to keep refining indefinitely, since after a finite amount of iterations the finest semi-partition $\{\ \{s\} \mid s \in S\ \}$ will be obtained. Hence we conclude the algorithm terminates. However, the size of a semi-partition of S can be exponential in $|S|$. Since after each iteration of the algorithm the number of blocks of the semi-partition is increased by at most one, the number of iterations may be exponential in $|S|$ as well. This is a huge decrease in performance from the original Groote-Vaandrager algorithm, which terminates after at most $|S| - 1$ iterations.

In the theorem below, we call a semi-partition $\mathcal{C}$ a *stable* partition with respect to a block B' if for no block B and for no featured label (α, ψ), B' is a splitter of B with respect to (α, ψ). The semi-partition $\mathcal{C}$ is itself called stable if $\mathcal{C}$ is stable with respect to all its blocks.

Lemma 6.2. *For an FTS fts* $= (S, \theta, s_*)$*,* $\mathcal{B}_{min}$ *obtained from the algorithm is the coarsest stable semi-partition refining* $\{S\}$*.*

Proof. We show by induction on the number of iterations of the algorithm that each stable partition refines the current semi-partition $\mathcal{B}$. Let $\mathcal{C}$ be a stable semi-partition. Clearly the statement holds initially, each semi-partition refines $\{S\}$. Suppose $\mathcal{C}$ refines semi-partition $\mathcal{B}$ obtained after a number of iterations and suppose a splitter B' of a block B exists with respect to a featured label (α, ψ). It suffices to show that any block C of $\mathcal{C}$ is included in a block of $\mathcal{B}'$, the semi-partition obtained by splitting B. Pick a block of $\mathcal{B}$ containing C. If this block is different from B, we are done. So, suppose $C \subseteq B$. We have to show that either $C \subseteq \mathit{non\text{-}neg}_{(\alpha,\psi)}(B, B')$ or $C \subseteq B \setminus \mathit{pos}_{(\alpha,\psi)}(B, B')$.

Suppose $s, t \in C$ with $s \in \mathit{pos}_{(\alpha,\psi)}(B, B')$ and $t \in \mathit{neg}_{(\alpha,\psi)}(B, B')$. We derive a contradiction. Pick a product $P \in \mathcal{P}$ such that $P \models \psi$. Such a product exists by definition of $\mathcal{A}_f$. Choose $s_0, \ldots, s_n \in B$, $s' \in B'$, $\psi_1, \ldots, \psi_n, \psi' \in \mathbb{B}(\mathcal{F})$ such that $s_0 = s$, $s_{i-1} \xrightarrow{\tau|\psi_i} s_i$ for $1 \leq i \leq n$, $s_n \xrightarrow{(\alpha|\psi')} s'$, and moreover $P \models \psi_i$, for $1 \leq i \leq n$, and $P \models \psi'$. Let $C_0, \ldots, C_n, C'$ be the blocks of $\mathcal{C}$ such that $s_i \in C_i$ and $s' \in C'$. Note that $C_i \subseteq B$, for $0 \leq i \leq n$, and $C' \subseteq B'$. Using the fact that $\mathcal{C}$ is stable we can construct a sequence $t_0, \ldots, t_m \in B$, $t' \in B'$, $\varphi_1, \ldots, \varphi_m, \varphi' \in \mathbb{B}(\mathcal{F})$ such that $t_0 = t$, $t_{i-1} \xrightarrow{\tau|\varphi_i} t_i$ for $1 \leq i \leq m$, $t_n \xrightarrow{(\alpha|\varphi')} t'$, and moreover $P \models \varphi_i$ for $1 \leq i \leq m$, and $P \models \varphi'$. This contradicts $t \in \mathit{neg}_{(\alpha,\psi)}(B, B')$, and proves the induction step. □

Given an FTS *fts* with state space S, we would now like to show that the semi-partition $\mathcal{B}_{min}$ obtained from the algorithm is equivalent to the partition $\langle S \rangle \underline{\leftrightarrow}_{cbf}$ induced by coherent branching feature bisimulation, as this would imply that a coherent branching feature bisimulation quotient is being calculated. However the example given in Figure 6.1 shows that this is not the case.

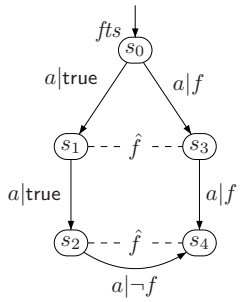


Figure 6.1: An FTS *fts* of which states s_1 and s_3 are coherent feature bisimilar. However, these states will not end up in the same block of the partition after applying the algorithm.

Example. Figure 6.1 shows an FTS *fts* of which states s_1 and s_3 are coherent feature bisimilar. This is witnessed by a feature bisimulation relation relating states s_2 and s_4 for the feature expression f. However, states s_2 and s_4 are not coherent feature bisimilar, since they are both reachable for all products, and only state s_2 can perform an a-step for the products satisfying $\neg f$. Hence, the algorithm will separate the states s_2 and s_4 on the featured label $(a, \neg f)$. As a result, this pair of states cannot witness that states s_1 and s_3 are in fact coherent feature bisimilar. Therefore, in a next iteration of the algorithm the pair of states s_1 and s_3 will be separated as well.

This example shows that the algorithm can only identify states as coherent feature bisimilar if this is witnessed by states that are also coherent feature bisimilar. We therefore define a slight adaptation of coherent branching feature bisimulation, which we refer to as *complete coherent branching feature bisimulation.*

Definition 6.1. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, and let ϱ denote its reachability function. We say two states $s, t \in S$ are complete coherent branching feature bisimilar, denoted by $fts \models s \underline{\leftrightarrow}_{ccbf} t$, if and only if there exists a branching feature bisimulation relation R on fts such that $(s, [\varrho(s) \wedge \varrho(t)]_{\sim_\mathcal{P}}, t) \in R$, and such that $\forall (s_1, \hat{\varphi}, s_2) \in R\colon \varrho(s_1) \wedge \varrho(s_2) \Rightarrow_\mathcal{P} \varphi$.*

Hence, two states are complete coherent branching feature bisimilar if and only if they are coherent branching feature bisimilar, and this fact is witnessed by a branching feature bisimulation relation containing only coherent branching feature bisimilar pairs of states. It follows that complete coherent branching feature bisimilarity is a stronger notion than coherent branching feature bisimilarity.

We show that the semi-partition $\mathcal{B}_{min}$ obtained from the algorithm is equal to the semi-partition $\langle S \rangle \underline{\leftrightarrow}_{ccbf}$ induced by complete coherent branching feature bisimulation. Note that, as complete coherent branching feature bisimulation is a stronger notion than coherent branching feature bisimulation, we have that $\langle S \rangle \underline{\leftrightarrow}_{ccbf}$ is a refinement of $\langle S \rangle \underline{\leftrightarrow}_{cbf}$. Hence, showing that $\mathcal{B}_{min}$ is equal to $\langle S \rangle \underline{\leftrightarrow}_{ccbf}$ implies that the complete coherent quotient obtained by post-processing $\mathcal{B}_{min}$ is a refinement of a coherent quotient of *fts*.

Theorem 6.2. *Assume that $\mathcal{B}_{min}$ is the partition obtained upon termination after applying the algorithm to the FTS $fts = (S, \mathcal{A}, \theta, s_*)$. Then $\mathcal{B}_{min} = \langle S \rangle \underline{\leftrightarrow}_{ccbf}$.*

Proof. Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS. We prove the theorem by showing that 1) $\mathcal{B}_{min}$ is a refinement of $\langle S \rangle \underline{\leftrightarrow}_{ccbf}$, and 2) that $\langle S \rangle \underline{\leftrightarrow}_{ccbf}$ is a refinement of $\mathcal{B}_{min}$.

We show that $\mathcal{B}_{min}$ is a refinement of $\langle S \rangle \underline{\leftrightarrow}_{ccbf}$ by showing that the relation

$$R = \{(s, [\varrho(s) \wedge \varrho(t)]_{\sim_\mathcal{P}}, t) \in S \times \widehat{\mathbb{B}}(\mathcal{F}) \times S \mid [s]_{\simeq_{\mathcal{B}_{min}}} = [t]_{\simeq_{\mathcal{B}_{min}}}\}$$

is a branching feature bisimulation relation. By construction of R, this implies

$$([s]_{\simeq_{\mathcal{B}_{min}}} = [t]_{\simeq_{\mathcal{B}_{min}}}) \Rightarrow fts \models s \underline{\leftrightarrow}_{ccbf} t,$$

for all states $s, t \in S$, which proves that $\mathcal{B}_{min}$ is a refinement of $\langle S \rangle \underline{\leftrightarrow}_{ccbf}$.

We show that R satisfies the transfer condition from Definition 4.13. Take a tuple of the form $(s, [\varrho(s) \wedge \varrho(t)]_{\sim_\mathcal{P}}, t) \in S \times \mathbb{B}(\mathcal{F}) \times S$, such that $[s]_{\simeq_{\mathcal{B}_{min}}} = [t]_{\simeq_{\mathcal{B}_{min}}} = B$. Suppose $s \xrightarrow{\alpha|\psi} s'$, for some $\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$ and $s' \in S$. This implies that $s \in pos_{(\alpha,\psi)}(B, B')$, where $B' = [s']_{\simeq_{\mathcal{B}_{min}}}$. Since $[t]_{\simeq_{\mathcal{B}_{min}}} = B$ and $\mathcal{B}_{min}$ is stable, by Theorem 6.2, it follows that $t \in non\text{-}neg_{(\alpha,\psi)}(B, B')$. Hence, by definition of *non-neg*, we have

$$\forall P \in \mathcal{P},\, P \models \varrho(t) \wedge \psi \colon \exists n\, \exists t_0, \ldots, t_n \in B\, \exists t' \in B'\, \exists \psi_1, \ldots, \psi_n, \psi' \colon$$
$$t_0 = t \wedge (\forall i, 1 \leq i \leq n \colon t_{i-1} \xrightarrow{\tau|\psi_i} t_i \wedge P \models \psi_i) \wedge t_n \xrightarrow{(\alpha|\psi')} t' \wedge P \models \psi' \,\},$$

By definition of R we have $(s, [\varrho(s) \wedge \varrho(t_i)]_{\sim_\mathcal{P}}, t_i) \in R$, for all $0 \leq i \leq n$, and $(s', [\varrho(s') \wedge \varrho(t')]_{\sim_\mathcal{P}}, t') \in R$, which satisfies the transfer condition.

We show that $\langle S \rangle \underline{\leftrightarrow}_{ccbf}$ is a refinement of $\mathcal{B}_{min}$ by showing that $\langle S \rangle \underline{\leftrightarrow}_{ccbf}$ is a stable semi-partition refining $\{S\}$.

By definition $\langle S \rangle \underline{\leftrightarrow}_{ccbf}$ is a semi-partition. Hence it remains to show that $\langle S \rangle \underline{\leftrightarrow}_{ccbf}$ is stable indeed. Suppose that there are blocks B, B' in $\langle S \rangle \underline{\leftrightarrow}_{ccbf}$ such that B' is a splitter of B with respect to a featured label (α, ψ). This means there are states s and t in B such that $s \in pos_{(\alpha,\psi)}(B, B')$ and $t \in neg_{(\alpha,\psi)}(B, B')$. We pick $P \in \mathcal{P}$ such that $P \models \varrho(s) \wedge \varrho(t) \wedge \psi$. Note that such P exists: by definition of featured labels we have $\hat{\psi} \neq \widehat{\mathsf{false}}$. By definition of *pos* we have $\psi \Rightarrow_\mathcal{P} \varrho(s)$, and by definition of *neg* we have $[\varrho(t) \wedge \psi]_{\sim_\mathcal{P}} \neq \widehat{\mathsf{false}}$.

By definition of the *pos*-set there exist $s_0, \ldots, s_n \in B$, $s' \in B'$, $\psi_1, \ldots, \psi_n, \psi' \in \mathbb{B}(\mathcal{F})$ such that $s_0 = s$, $s_{i-1} \xrightarrow{\tau|\psi_i} s_i$ for $1 \leqslant i \leqslant n$, $s_n \xrightarrow{(\alpha|\psi')} s'$, and moreover $P \models \psi_i$, for $1 \leqslant i \leqslant n$, and $P \models \psi'$.

Since s_n and t are in the same block of $\langle S \rangle \underline{\leftrightarrow}_{ccbf}$ we have $fts \models s_n \,\underline{\leftrightarrow}_{ccbf}\, t$. Let R be the bisimulation relation that witnesses this. Using the transfer condition of this relation we can construct a sequence $t_0, \ldots, t_m \in B$, $t' \in B'$, $\varphi_1, \ldots, \varphi_m, \varphi' \in \mathbb{B}(\mathcal{F})$ such that $t_0 = t$, $t_{i-1} \xrightarrow{\tau|\varphi_i} t_i$ for $1 \leqslant i \leqslant m$, $t_n \xrightarrow{(\alpha|\varphi')} t'$, and moreover $P \models \varphi_i$ for $1 \leqslant i \leqslant m$, and $P \models \varphi'$. This contradicts $t \in neg_{(\alpha,\psi)}(B, B')$, and proves that $\langle S \rangle \underline{\leftrightarrow}_{ccbf}$ is stable.

Since $\mathcal{B}_{min}$ is the coarsest stable semi-partition refining $\{S\}$ by Theorem 6.2, we can conclude that $\langle S \rangle \underline{\leftrightarrow}_{ccbf}$ is a refinement of $\mathcal{B}_{min}$.

□

Thus, given an FTS *fts* with set of states S, we continue to refine the trivial semi-partition until no more splitters can be found. The final semi-partition $\mathcal{B}_{min}$ that is reached is equal to the semi-partition induced on S by complete coherent branching feature bisimulation. A complete coherent branching feature bisimulation quotient can now be constructed by finding a smallest partition $\mathcal{C}$ of S that refines $\mathcal{B}_{min}$. This last step requires solving the set cover problem, which is NP-hard.

6.3 Implementation details

The implementation of the algorithm described in this section assumes its input is already pre-processed, i.e. the reachability information of the states is already calculated and pushed on the transitions, and all unreachable elements are removed.

The algorithm uses three defined data types: blocks, states, and transitions. The initial semi-partition P_0 contains a single block B. this block contains all states of the FTS. It moreover has a list with its incoming inert transitions, which are all transitions labeled with action τ, and a list containing its incoming non-inert transitions, which are all transitions labeled with an action other than τ.

States contain a list of all their incoming transitions, a list of blocks they appear in (initially $[B]$ for all states), a set of products for which they are reachable, and a product set that is used as a flag, which is initially empty.

Each transition contains the state it comes from, the state it leads to, the action it is labeled with and the set of products it is labeled with.

Formal descriptions of these data types can be found in Appendix D. We proceed by discussing the pseudo code of the algorithm, which is shown below.

Algorithm *BFB-Reduction*(P_0)
1. **list** *toBeProcessed* := P_0
2. **list** *Stable* := $\emptyset$
3. **for** $B' \in$ *toBeProcessed*
4. **for** $(\alpha, \varphi) \in B'.labels$
5. **list** $BL := \emptyset$
6. **for** $t \in B'.nonInert : t.label == \alpha$
7. *RaiseFlags*($\{t\}$)
8. $BL := BL \cup t.from.blocks$
9. **list** $pos[B \in BL] := \{s \in B.states \mid (s.products \cap \varphi) \setminus s.flag == \emptyset\}$
10. **list** $all[B \in BL] := \{s \in pos[B] \mid \varphi \subseteq s.products\}$
11. **for** $B \in BL : all[B] \neq \emptyset \wedge pos[B] \neq B$
12. *RemoveBlock*(B)
13. **if** $\forall b \in Blocks : pos[B] \not\subseteq b.States$
14. **then** *toBeProcessed.add*(*splitBlock*(B, $pos[B]$))
15. **if** $\forall b \in Blocks : B \setminus all \not\subseteq b.States$
16. **then** *toBeProcessed.add*(*splitBlock*(B, $B \setminus all[B]$))
17. *LowerFlags*(BL)
18. **if** $B' \in BL$
19. **then break**
20. **if** $B' \notin BL$
21. **then** *toBeProcessed.remove*(B')
22. *Stable.add*(B')
23. **return** *Stable*

The algorithm uses three lists of blocks: *toBeProcessed*, *Stable* and *BL*. The list *toBeProcessed* contains all blocks that might be a splitter of some other block. The list *Stable* contains all blocks that are proven to be stable. The last list is an auxiliary list used to store the blocks that might need splitting.

Initially, all states are contained in a single block (P_0), which is added to the list *toBeProcessed* (`line 1`). The *Stable* list is initially empty (`line 2`).

While the list *toBeProcessed* is not empty, a block B' is taken from it (`line 3`), and it is checked of which blocks it is a splitter with regard to a featured label (α, φ) incoming to B' (`lines 4-11`). This is done by raising the flags of all states that can reach B' with an α-transition (`lines 6-7`). All blocks containing such a state are added to the list BL (`line 7`). Hereafter the *pos*-set and *all*-set are calculated for each block in the list BL (`lines 9-10`).

The blocks of which B' is a splitter are split and newly created blocks are added to the list *toBeProcessed* (`lines 12-16`). The flags are reset before continuing to the next featured label (`line 17`). If B' itself was split, we continue immediately by evaluating the next block, instead of evaluating the next featured label (`lines 18-19`). If the splitting was performed with regard to all incoming featured labels, and B' itself was never split, it is added to the list *Stable*, and we continue with the next block from *toBeProcessed* (`lines 20-22`).

When *toBeProcessed* is empty, the list of stable blocks is returned (`line 23`), which at this point contains the coarsest stable semi-partition. Post-processing has to be applied to this data to obtain the state space of a complete coherent branching feature bisimulation quotient.

Pseudo code of the subroutines is included in Appendix D.

We conclude this chapter by summarizing the achieved results. Our goal was to define an algorithm that calculates a coherent branching feature bisimulation quotient of a given FTS. In Section 6.1 we established that this problem is NP-hard. Hence, we are not able to solve it efficiently.

In Section 6.2 we presented our adaptation of the Groote-Vaandrager algorithm. We showed that both the pre-processing and post-processing steps of the algorithm require to solve NP-hard problems. Furthermore, the size of the semi-partition resulting from the main algorithm may be exponential in the number of states, and hence this part of the algorithm may also take exponential time. We found that the presented algorithm may not compute a coherent branching feature bisimulation quotient. Instead a refinement is calculated, which we refer to as the complete coherent branching feature bisimulation quotient. We formally proved that this refinement is indeed calculated by the algorithm.

As we now have an algorithm to perform branching feature bisimulation reduction on event-based family-level models, it would be beneficial if this algorithm is applicable to state-based family-level models as well. Hence, in the next chapter we will investigate the possibility of converting from one type of model to the other and back, in order to facilitate this.

Chapter 7

Minimization using embeddings

As discussed earlier, state space reduction is a major application for equivalence relations on both state-based and event-based models. For this purpose we defined several quotients in Chapter 5 using the notions of (branching) feature bisimulation for both FKS and FTS. However, as we have seen in the previous chapter, designing and implementing an algorithm to calculate such a quotient, in either world, is a challenging task. Hence it would be beneficial if existing algorithms could be used to perform minimization in both worlds, regardless of the world they were designed for.

In this chapter, we will discuss the possibilities of minimizing state-based models by performing minimization for event-based models, and the other way around. We restrict ourselves to minimization modulo feature bisimulation. We leave minimization modulo branching feature bisimulation / divergence-blind stuttering feature equivalence to future work.

In order to compare the two worlds, we first define transformations, called *embeddings*, from state-based models to event-based models, and from event-based models to state-based models. Using these embeddings, we are able to compare the naive and coherent feature bisimulation quotients in both worlds. In the end, we are able to prove that it is possible to minimize a model by minimizing its embedding in the other world.

7.1 Minimization for state-based models

In this section we discuss the possibilities of minimizing state-based models by performing minimization for event-based models. Sections 7.1.1 and 7.1.2 discuss embeddings from state-based to event-based models, and Section 7.1.4 presents the possibilities for minimization using these embeddings.

7.1.1 Embeddings to event-based models

In [14] an embedding from KS to LTS has been proposed. The definition of this embedding is given below.

Definition 7.1. *Let* $ks = (S, AP, \rightarrow_{ks}, L, s_*)$ *be a KS. The embedding* $\texttt{lts}\colon \texttt{KS} \rightarrow \texttt{LTS}$ *is defined such that* $\texttt{lts}(ks) = (S', \mathcal{A}, \rightarrow_{lts}, s_*)$ *is an LTS such that:*

- $S' = S \cup \{\bar{s} \mid s \in S\}$, *such that* $\bar{s} \notin S$ *for all* $s \in S$,
- $\mathcal{A} = 2^{AP} \cup \{\bot\}$,
- $\rightarrow_{lts} \subseteq S' \times \mathcal{A}_\tau \times S'$ *is the least relation satisfying the following rules for all* $s, t \in S$*:*

$$\frac{}{s \xrightarrow{\bot}_{lts} \bar{s}} \qquad \frac{}{\bar{s} \xrightarrow{L(s)}_{lts} s} \qquad \frac{s \rightarrow_{ks} t \qquad L(s) = L(t)}{s \xrightarrow{\tau}_{lts} t} \qquad \frac{s \rightarrow_{ks} t \qquad L(s) \neq L(t)}{s \xrightarrow{L(t)}_{lts} t}$$

In this embedding each state is duplicated in order to facilitate the encoding of the state information of the KS. The fresh symbol $\bot$ is used to signal a forthcoming of such information. An example of the application of the embedding `lts` is shown in Figure 7.1.

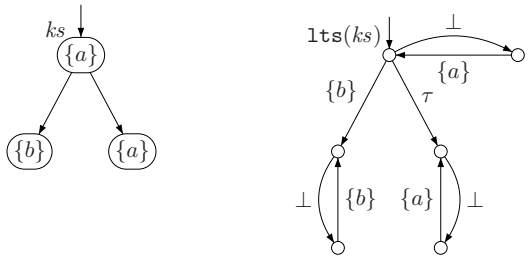


Figure 7.1: A KS *ks* and its corresponding LTS `lts`(*ks*).

Example. In Figure 7.1 the KS *ks* is embedded into the LTS `lts`(*ks*). The three states of *ks* are duplicated, and transitions between the original and duplicated states are added, labeled with $\bot$ (from the original to the duplicate), and with the state information (from the duplicate to the original). The two transitions of *ks* are in `lts`(*ks*) labeled with the state information of the target state in case of a non-silent transition, and with τ in case of a silent transition.

In [26], it has been proven that the embedding `lts` preserves and reflects bisimulation.

Theorem 7.1. *Let* $ks = (S, AP, \rightarrow, L, s_*)$ *be a KS. Then, for all* $s, s' \in S$, $ks \models s \leftrightarrows s'$ *if and only if* $\texttt{lts}(ks) \models s \leftrightarrows s'$.

Proof. As in Theorem 5.3 of [26]. □

With the embedding of KS into LTS given, the challenge here is to find an embedding from state-based models to event-based models on the product family level. We first propose a straightforward generalization from the embedding `lts` to an embedding `fts`, that transforms an FKS to an FTS.

Definition 7.2. *Let* $fks = (S, AP, \theta, L, s_*)$ *be an FKS. The embedding* $\texttt{fts}: \texttt{FKS} \rightarrow \texttt{FTS}$ *is defined such that* $\texttt{fts}(fks) = (S', \mathcal{A}, \theta', s_*)$ *is an FTS such that:*

- $S' = S \cup \{\bar{s} \mid s \in S\}$, *such that* $\bar{s} \notin S$ *for all* $s \in S$,

- $\mathcal{A} = 2^{AP} \cup \{\bot\}$,
- $\theta' : S' \times \mathcal{A}_\tau \times S' \to \mathbb{B}(\mathcal{F})$ *is constructed such that, for all* $s, t \in S'$ *and* $\alpha \in \mathcal{A}_\tau$*:*

$$\theta'(s,\alpha,t) = \begin{cases} \mathsf{true} & \text{if } s \in S \wedge t = \bar{s} \wedge \alpha = \bot \\ \mathsf{true} & \text{if } t \in S \wedge s = \bar{t} \wedge \alpha = L(t) \\ \theta(s,t) & \text{if } s,t \in S \wedge L(s) = L(t) \wedge \alpha = \tau \\ \theta(s,t) & \text{if } s,t \in S \wedge L(s) \neq L(t) \wedge \alpha = L(t) \\ \mathsf{false} & \text{otherwise} \end{cases}$$

This embedding is, apart from the feature-expressions attached to the transitions, identical to the embedding `lts`. The newly-introduced transitions that facilitate the encoding of the state information of the FKS are assigned the feature expression `true`. The feature-expressions attached to the transitions in the FKS are carried over unaltered to the transitions in the FTS. An example of the application of the embedding `fts` is shown in Figure 7.2.

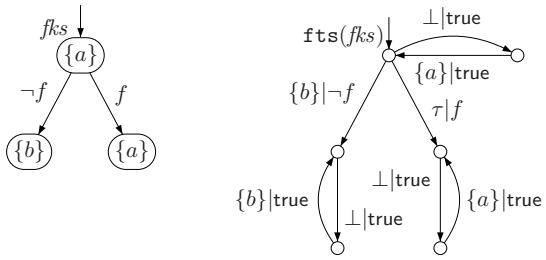


Figure 7.2: An FKS *fks* and its corresponding FTS `fts`(*fks*).

For this embedding we can prove that projection on a product distributes over the embedding from a state-based to an event-based model. This lemma will be useful to lift properties already proven for the embeddings on the single-product level to the product-family level. The proof for the lemma can be found in Appendix E.

Lemma 7.1. *Let fks be an FKS. Then, for all* $P \in \mathcal{P}$*, we have* $\mathtt{fts}(fks)|_P = \mathtt{lts}(fks|_P)$.

Using this lemma we prove that the embedding `fts` preserves and reflects feature bisimulation, using that the embedding `lts` preserves and reflects strong bisimulation. However, reflection can only be proven under the additional assumption that the labels of the related states are equal. This is due to the additional constraint in Theorem 4.2, stating that bisimulation on the single-product level of a state-based model can only be lifted to feature bisimulation on the product-family level if the state-labeling is equal.

Theorem 7.2. *Let* $fks = (S, AP, \theta, L, s_*)$ *be an FKS. Then, for all* $s, s' \in S$*, and* $\varphi \in \mathbb{B}(\mathcal{F})$*, we have* $fks \models s \overset{\varphi}{\leftrightarrow}_f s'$ *if and only if* $\mathtt{fts}(fks) \models s \overset{\varphi}{\leftrightarrow}_f s' \wedge L(s) = L(s')$.

Proof. Let $fks = (S, AP, \theta, L, s_*)$ be an FKS, and let $s, s' \in S$ and $\varphi \in \mathbb{B}(\mathcal{F})$.

Suppose $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f s'$. By Theorem 4.1 this implies $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathit{fks}|_P \models s \leftrightarrows s'$. By Theorem 7.1 we get $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathtt{lts}(\mathit{fks}|_P) \models s \leftrightarrows s'$, and by Lemma 7.1 this is equivalent to $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathtt{fts}(\mathit{fks})|_P \models s \leftrightarrows s'$. Now we use Theorem 4.6 to obtain $\mathtt{fts}(\mathit{fks}) \models s \overset{\varphi}{\leftrightarrows}_f s'$. Furthermore, by Definition 4.2 it immediately follows that $L(s) = L(s')$.

Suppose $\mathtt{fts}(\mathit{fks}) \models s \overset{\varphi}{\leftrightarrows}_f s'$. By Theorem 4.5 this implies $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathtt{fts}(\mathit{fks})|_P \models s \leftrightarrows s'$, which is equivalent to $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathtt{lts}(\mathit{fks}|_P) \models s \leftrightarrows s'$ by Lemma 7.1. Using Theorem 7.1 we obtain $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathit{fks}|_P \models s \leftrightarrows s'$, from which we conclude that $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f s'$, by Theorem 4.2, using that $L(s) = L(s')$. □

Although this embedding does preserve bisimulation, we still found it to be unsuitable for our purposes. Consider the FKS *fks* shown in Figure 7.3. Note that *fks* is minimal modulo strong feature bisimulation.

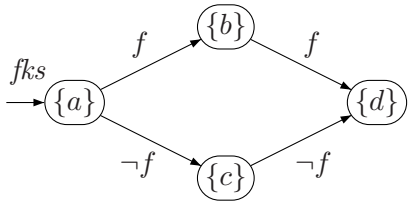


Figure 7.3: Example FKS *fks* of which minimality modulo coherent feature bisimulation is not preserved after applying the embedding `fts`.

The FTS $\mathtt{fts}(\mathit{fks})$ resulting from embedding *fks* is shown in Figure 7.4. One of the coherent quotients of $\mathtt{fts}(\mathit{fks})$ is shown in Figure 7.5.

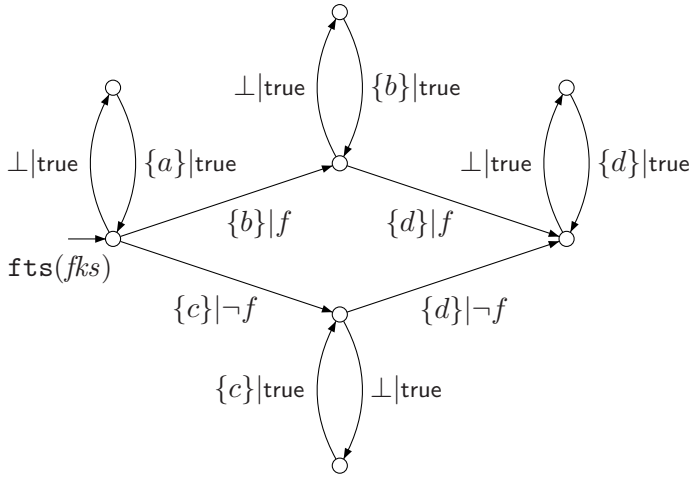


Figure 7.4: Embedding $\mathtt{fts}(\mathit{fks})$ of the FKS *fks* from Figure 7.3

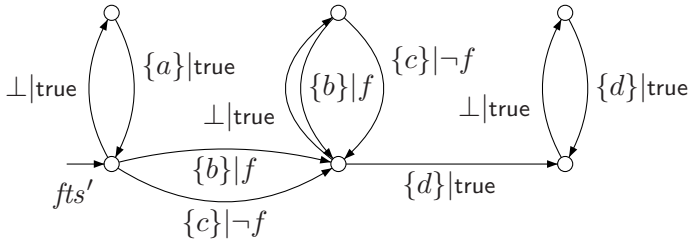


Figure 7.5: Coherent quotient fts' of the FTS $\texttt{fts}(fks)$ from Figure 7.4

The quotient from Figure 7.5 has fewer states than the FTS $\texttt{fts}(fks)$, proving that $\texttt{fts}(fks)$ is not minimal modulo coherent feature bisimulation. However, the FKS *fks* is minimal modulo strong feature bisimulation, and hence also modulo coherent feature bisimulation, since all states have a different label. So although it may be possible, with some creativity, to revert this quotient back to an FKS (which would then be equal to the original FKS), this is not what we are looking for; preservation of minimality is a nice property for an embedding, and hence we would like our embeddings to possess this property.

Note that this embedding is actually suitable for naive feature bisimulation reduction. That is, the embedding `fts` does preserve minimality modulo naive feature bisimulation. However, we are looking for a more generic embedding that is compatible with both naive- and coherent feature bisimulation reduction.

Hence, we will propose an alternative embedding in the next section.

7.1.2 Embeddings$^+$ to event-based models

The previously proposed embedding `fts` was not suitable for coherent bisimulation reduction in FKS, since bisimulation for FTS does not have a constraint that is equivalent to the constraint on FKS that the labels of two states have to be equal in order for those states to be feature bisimilar. Hence, coherent bisimulation reduction on FTS can group states together that are not feature bisimilar in the original FKS.

To solve this problem, we propose an adapted definition of the embedding `fts`. In this new definition we add a dummy feature to the set of features, and a dummy product to the set of products. This dummy product is used in the FTS to make sure the labels of states are equal in the FKS. We call this new embedding the embedding$^+$, since it is designed to always work with at least one product: the newly introduced dummy product. Since we will be modifying the feature- and product sets with this embedding, we will now consider these sets as part of FTS and FKS definitions.

Some of the theorems in this section are very similar to the ones in Section 7.1.1, and hence the proofs are omitted. The interested reader is refered to Appendix E, as well as for the proofs of the lemmas in this section.

Definition 7.3. *Let* $fks = (S, AP, \theta, L, s_*, \mathcal{F}, \mathcal{P})$ *be an FKS. The embedding* $\texttt{fts}^+ : \texttt{FKS} \to \texttt{FTS}$ *is defined such that* $\texttt{fts}^+(fks) = (S', \mathcal{A}, \theta', s_*, \mathcal{F}', \mathcal{P}')$ *is an FTS such that:*

- $S' = S \cup \{\bar{s} \mid s \in S\}$, *such that* $\bar{s} \notin S$ *for all* $s \in S$,

- $\mathcal{A} = 2^{AP} \cup \{\bot\}$,
- $\theta' : S' \times \mathcal{A}_\tau \times S' \to \mathbb{B}(\mathcal{F}')$ *is constructed such that, for all* $s, t \in S'$ *and* $\alpha \in \mathcal{A}_\tau$*:*

$$\theta'(s,\alpha,t) = \begin{cases} \mathsf{true} & \text{if } s \in S \wedge t = \bar{s} \wedge \alpha = \bot \\ \mathsf{true} & \text{if } t \in S \wedge s = \bar{t} \wedge \alpha = L(t) \\ \theta(s,t) \wedge \neg f_\bot & \text{if } s,t \in S \wedge L(s) = L(t) \wedge \alpha = \tau \\ \theta(s,t) \wedge \neg f_\bot & \text{if } s,t \in S \wedge L(s) \neq L(t) \wedge \alpha = L(t) \\ \mathsf{false} & \text{otherwise} \end{cases}$$

- $\mathcal{F}' = \mathcal{F} \cup \{f_\bot\}$, *such that* $f_\bot \notin \mathcal{F}$,
- $\mathcal{P}' = \mathcal{P} \cup \{\{f_\bot\}\}$.

In this adaptation of the embedding `fts`, all transitions facilitating the encoding of the state-information of the FKS are still assigned the feature expression true, which means they are also enabled for the dummy product $\{f_\bot\}$. The constraint $\neg f_\bot$ is added to the feature expressions of all other transitions, meaning that these transitions are not enabled for the dummy product. An example of an application of the embedding $\texttt{fts}^+$ is shown in Figure 7.6.

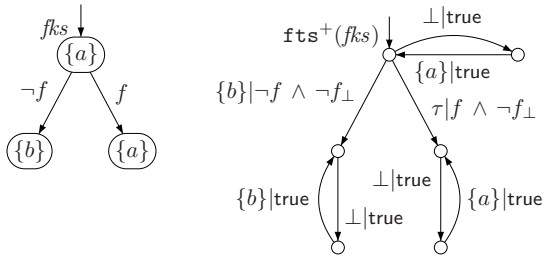


Figure 7.6: An FKS *fks* and its corresponding FTS $\texttt{fts}^+(\mathit{fks})$.

We show that bisimilarity of two states in an FKS corresponds to bisimilarity of their respective additional states in the embedded FKS using the $\texttt{fts}^+$ embedding.

Lemma 7.2. *Let* $\mathit{fks} = (S, AP, \theta, L, s_*, \mathcal{F}, \mathcal{P})$ *be an FKS. Then we have, for all* $s, s' \in S$, $\texttt{fts}^+(\mathit{fks}) \models s \overset{\varphi}{\leftrightarrows}_f s' \Leftrightarrow \texttt{fts}^+(\mathit{fks}) \models \bar{s} \overset{\varphi}{\leftrightarrows}_f \bar{s}'$, *for all* $\varphi \in \mathbb{B}(\mathcal{F})$.

Similar to the embedding `fts`, the embedding $\texttt{fts}^+$ also distributes over projection.

Lemma 7.3. *Let fks be an FKS. Then, for all* $P \in \mathcal{P}$, *we have* $\texttt{fts}^+(\mathit{fks})|_P = \texttt{lts}(\mathit{fks}|_P)$.

In Theorem 7.2 we showed that the embedding `fts` only reflects feature bisimilarity if the labels of the related states are equal. We will show this constraint coincides with states being $f_\bot$-bisimilar in the embedding $\texttt{fts}^+$.

Lemma 7.4. *Let* $\mathit{fks} = (S, AP, \theta, L, s_*, \mathcal{F}, \mathcal{P})$ *be an FKS. Then we have, for all* $s, s' \in S$, $\texttt{fts}^+(\mathit{fks})|_{\{f_\bot\}} \models s \leftrightarrows s' \Leftrightarrow L(s) = L(s')$.

Using the above lemma, we can formulate the additional constraint for reflection of feature bisimilarity in an alternative way.

Theorem 7.3. *Let* $fks = (S, AP, \theta, L, s_*, \mathcal{F}, \mathcal{P})$ *be an FKS. Then, for all* $s, s' \in S$, *and* $\varphi \in \mathbb{B}(\mathcal{F})$, *we have* $fks \models s \overset{\varphi}{\leftrightarrow}_f s'$ *if and only if* $\mathtt{fts}^+(fks) \models s \overset{\varphi \vee f_\perp}{\leftrightarrow}_f s'$.

Now that the additional constraint is formulated as part of the feature bisimulation constraint for the FTS, we can define a slightly adapted coherent quotient for FTS that respects this constraint. This quotient is discussed in Section 7.1.3.

As we will see in Section 7.1.4, this definition for the embedding does preserve minimality for both naive feature bisimulation and coherent feature bisimulation. Therefore we believe this embedding is suitable for our purposes, and hence we define the reverse embedding $\mathtt{fts}^{+-1}$. Since this reverse embedding will not be applicable to all FTS, we first specify which FTS are *reversible*. Note that properties 5*a*, 5*b*, 5*d*, 5*e* and 6*b* of the definition of reversible FTS are not strictly necessary for the embedding $\mathtt{fts}^{+-1}$ to be applicable. However, we will need those properties further on in Section 7.1.4 to prove that the embedding does preserve minimality.

Definition 7.4. *Let* $fts = (S, \mathcal{A}, \theta, s_*, \mathcal{F}, \mathcal{P})$ *be an FTS. Then fts is reversible iff the following conditions are satisfied:*

1. $\mathcal{A} = 2^{AP} \cup \{\perp\}$, *for some set* AP.
2. $f_\perp \in \mathcal{F}$ *and* $\{f_\perp\} \in \mathcal{P}$. *Furthermore,* $\forall P \in \mathcal{P}\colon (\, P \neq \{f_\perp\} \;\Rightarrow\; P \not\models f_\perp \,)$.
3. S *can be partitioned in two sets* $S_\perp$ *and* $\bar{S}$.
4. $s_* \in S_\perp$.
5. *For all* $s_\perp, t_\perp \in S_\perp$, $\bar{s}, \bar{t} \in \bar{S}$, *and* $\alpha, \alpha' \in \mathcal{A}_\tau$, *we require that* θ *is such that*
   (a) $(f_\perp \wedge \theta(s_\perp, \alpha, t_\perp)) \sim_{\mathcal{P}} \mathsf{false}$.
   (b) $\theta(s_\perp, \perp, \bar{s}) \not\sim_{\mathcal{P}} \mathsf{false}$ *implies* $f_\perp \Rightarrow_{\mathcal{P}} \theta(s_\perp, \perp, \bar{s})$.
   (c) $(f_\perp \vee \varrho(s_\perp)) \Rightarrow_{\mathcal{P}} \bigvee_{\bar{s} \in \bar{S}} \theta(s_\perp, \perp, \bar{s})$.
   (d) $\theta(\bar{s}, \alpha, \bar{t}) \sim_{\mathcal{P}} \mathsf{false}$.
   (e) $\theta(\bar{s}, \alpha, s_\perp) \not\sim_{\mathcal{P}} \mathsf{false}$ *and* $\theta(\bar{s}, \alpha', t_\perp) \not\sim_{\mathcal{P}} \mathsf{false}$ *implies* $\alpha = \alpha'$.
6. *For each* $s_\perp \in S_\perp$ *there exists an action* $L(s_\perp) \in \mathcal{A} \setminus \{\perp\}$ *such that, for all* $t_\perp \in S_\perp$, $\bar{s} \in \bar{S}$, *and* $\alpha \in \mathcal{A}_\tau$:
   (a) $\theta(\bar{s}, \alpha, s_\perp) \not\sim_{\mathcal{P}} \mathsf{false}$ *implies* $\alpha = L(s_\perp)$.
   (b) $\theta(\bar{s}, L(s_\perp), s_\perp) \sim_{\mathcal{P}} \theta(s_\perp, \perp, \bar{s})$.
   (c) $L(t_\perp) = L(s_\perp)$ *and* $\theta(t_\perp, \alpha, s_\perp) \not\sim_{\mathcal{P}} \mathsf{false}$ *implies* $\alpha = \tau$.
   (d) $L(t_\perp) \neq L(s_\perp)$ *and* $\theta(t_\perp, \alpha, s_\perp) \not\sim_{\mathcal{P}} \mathsf{false}$ *implies* $\alpha = L(s_\perp)$.

Note that any embedding $\mathtt{fts}^+(fks)$ of an FKS *fks* is a reversible FTS. We now give the definition of the reverse embedding $\mathtt{fts}^{+-1}$.

Definition 7.5. *Let $fts = (S, \mathcal{A}, \theta, s_*, \mathcal{F}, \mathcal{P})$ be a reversible FTS. The embedding $\texttt{fts}^{+-1}\colon \texttt{FTS} \to \texttt{FKS}$ is defined such that $\texttt{fts}^{+-1}(fts) = (S', AP, \theta', L, s_*, \mathcal{F}', \mathcal{P}')$ is an FKS such that:*

- $S' = \{s \in S \mid s \xrightarrow{\perp}\}$,
- *AP is such that $\mathcal{A} = 2^{AP} \cup \{\perp\}$,*
- *For all $s \in S'$, $L(s) = a$, for the unique $a \in 2^{AP}$ such that $s' \xrightarrow{a} s$, for some $s' \in S$.*
- *$\theta'\colon S' \times S' \to \mathbb{B}(\mathcal{F})$ is constructed such that, for all $s, t \in S'$,*

$$\theta'(s,t) = \theta(s, L(t), t) \vee \theta(s, \tau, t).$$

- $\mathcal{F}' = \mathcal{F} \setminus \{f_\perp\}$.
- $\mathcal{P}' = \mathcal{P} \setminus \{\{f_\perp\}\}$.

We show that the defined reverse embedding $\texttt{fts}^{+-1}$ is the left inverse of the embedding $\texttt{fts}^{+}$, by proving that composing these embeddings results in the identify function.

Theorem 7.4. *We have $\texttt{fts}^{+-1} \circ \texttt{fts}^{+} = \texttt{Id}$.*

Proof. Let $fks = (S, AP, \theta, L, s_*, \mathcal{F}, \mathcal{P})$ be an FKS. Let $\texttt{fts}^{+}(fks) = (S', \mathcal{A}', \theta', s'_*, \mathcal{F}', \mathcal{P}')$ and $\texttt{fts}^{+-1}(\texttt{fts}^{+}(fks)) = (S'', AP', \theta'', L', s''_*, \mathcal{F}'', \mathcal{P}'')$. We establish the isomorphism by proving that there are isomorphisms between S and S'', AP and AP', θ and θ'', L and L', s_* and s''_*, $\mathcal{F}$ and $\mathcal{F}''$, and $\mathcal{P}$ and $\mathcal{P}''$.

From the definition of $\texttt{fts}^{+}$ (applied to *fks*) it follows that

- $S' = S \cup \{\bar{s} \mid s \in S\}$,
- $\mathcal{A}' = 2^{AP} \cup \{\perp\}$,
- For $s, t \in S'$ and $\alpha \in \mathcal{A}'_\tau$ we have:

$$\theta'(s,\alpha,t) = \begin{cases} \mathsf{true} & \text{if } s \in S \wedge t = \bar{s} \wedge \alpha = \perp \\ \mathsf{true} & \text{if } t \in S \wedge s = \bar{t} \wedge \alpha = L(t) \\ \theta(s,t) \wedge \neg f_\perp & \text{if } s, t \in S \wedge L(s) = L(t) \wedge \alpha = \tau \\ \theta(s,t) \wedge \neg f_\perp & \text{if } s, t \in S \wedge L(s) \neq L(t) \wedge \alpha = L(t) \\ \mathsf{false} & \text{otherwise} \end{cases}$$

- $s'_* = s_*$.
- $\mathcal{F}' = \mathcal{F} \cup \{f_\perp\}$.
- $\mathcal{P}' = \mathcal{P} \cup \{\{f_\perp\}\}$.

An application of $\texttt{fts}^{+-1}$, applied to $\texttt{fts}^{+}(fks)$, gives:

- $S'' = \{s' \in S' \mid s' \xrightarrow{\perp}\} = \{s' \in (\, S \cup \{\bar{s} \mid s \in S\}\,) \mid s' \xrightarrow{\perp}\}$. Since $s' \xrightarrow{\perp}$ iff $s' \in S$, we obtain $S'' = \{s' \in S\} = S$.

- AP' is such that $\mathcal{A}' = 2^{AP'} \cup \{\bot\}$. Since $\mathcal{A}' = 2^{AP} \cup \{\bot\}$, we obtain $AP' = AP$.
- For all $s \in S''$, $L'(s) = a$ for the unique $a \in 2^{AP}$ such that $s' \xrightarrow{a} s$, for some $s' \in S''$. By construction of θ' we know that $s' = \bar{s}$ and $a = L(s)$. Hence, $L' = L$.
- Let $s, t \in S''$ be two states. Note that the images of s and t in S are s and t, respectively. We have:
$$\theta''(s,t) = \theta'(s, L'(t), t) \vee \theta'(s, \tau, t).$$
We now distinguish two cases.
  1. If $L(s) = L(t)$ we have $\theta'(s, L'(t), t) = \theta'(s, L(t), t) = \mathsf{false}$, and $\theta'(s, \tau, t) = \theta(s,t) \wedge \neg f_\bot$.
  2. Otherwise, if $L(s) \neq L(t)$, we have $\theta'(s, L'(t), t) = \theta'(s, L(t), t) = \theta(s,t) \wedge \neg f_\bot$, and $\theta'(s, \tau, t) = \mathsf{false}$.

  In both cases, we find that $\theta''(s,t) = \theta(s,t) \wedge \neg f_\bot$. Since $\theta(s,t) \wedge \neg f_\bot \sim_{\mathcal{P}} \theta(s,t)$ and s and t are there own images, we can conclude that $\theta'' = \theta$.
- $s''_* = s'_* = s_*$.
- $\mathcal{F}'' = \mathcal{F}' \setminus \{f_\bot\} = (\mathcal{F} \cup \{f_\bot\}) \setminus \{f_\bot\} = \mathcal{F}$.
- $\mathcal{P}'' = \mathcal{P}' \setminus \{\{f_\bot\}\} = (\mathcal{P} \cup \{\{f_\bot\}\}) \setminus \{\{f_\bot\}\} = \mathcal{P}$.

We conclude that $\mathtt{fts}^{+\,-1} \circ \mathtt{fts}^{+} = \mathtt{Id}$. □

Hence we have shown that we can transform and FKS *fks* to and from an event-based model using the embeddings $\mathtt{fts}^{+}$ and $\mathtt{fts}^{+\,-1}$ without changing *fks*. In the next section we discuss how the coherent quotient for FTS should be adapted in order to work with this embedding.

7.1.3 The coherent$^+$ quotients for FTS

With Lemma 7.4 we showed that two states of an FKS *fks* have the same label if and only if these states are $f_\bot$-bisimilar in the FTS $\mathtt{fts}^{+}(fks)$. Hence, a quotient for $\mathtt{fts}^{+}(fks)$ that respects $f_\bot$-bisimilarity does also respect the constraint that two equivalent states should have the same label in *fks*.

Including $f_\bot$ in the feature expression of the naive quotient is a simple method to make this quotient respect $f_\bot$-bisimilarity. That is, the $\overset{\phi}{\leftrightarrows}_f$-quotient for *fks* should correspond to the $\overset{\phi \vee f_\bot}{\leftrightarrows}_f$-quotient for $\mathtt{fts}^{+}(fks)$, for any feature expression $\phi \in \mathbb{B}(\mathcal{F})$. We will come back to this conjecture in the next section.

On the other hand, the coherent quotient from Definition 5.8 does not respect $f_\bot$-bisimilarity for $\mathtt{fts}^{+}(fks)$, since most states in this FTS are not reachable for the product $\{f_\bot\}$. Hence, the coherent quotient ignores this product. Therefore, we will provide an adaptation of the coherent quotient that always takes this product into account, whether it is reachable or not. In order to do this, we first adapt the definition of coherent feature bisimulation.

Definition 7.6. *Let $fts = (S, \mathcal{A}, \theta, s_*, \mathcal{F}, \mathcal{P})$ be an FTS such that $f_\perp \in \mathcal{F}$, and let ϱ denote its reachability function. We say two states $s, t \in S$ are coherent$^+$ feature bisimilar, denoted by $fts \models s \leftrightarrows_{cf+} t$ if and only if $fts \models s \xleftrightarrow{(\varrho(s) \wedge \varrho(t)) \vee f_\perp}_f t$.*

The coherent$^+$ feature bisimulation quotient is now defined as follows:

Definition 7.7. *Let $fts = (S, \mathcal{A}, \theta, s_*, \mathcal{F}, \mathcal{P})$ be an FTS such that $f_\perp \in \mathcal{F}$, with reachability function ϱ. A coherent$^+$ bisimulation quotient of fts is an FTS $fts' = (S', \mathcal{A}, \theta', s'_*, \mathcal{F}, \mathcal{P})$, such that*

- $S' \subseteq \bigcup_{C \in \langle S \rangle \underline{\leftrightarrows}_{cf+}} 2^C$, *such that*
  1. $\bigcup S' = S$;
  2. $C_1 \cap C_2 = \emptyset$, *for all distinct* $C_1, C_2 \in S'$;
  3. $|S'|$ *is minimal.*
- $\theta' \colon S' \times \mathcal{A}_\tau \times S'$ *is constructed such that, for all* $C_1, C_2 \in S'$ *and* $\alpha \in \mathcal{A}_\tau$*:*

$$\theta'(C_1, \alpha, C_2) = \bigvee \{\, \theta(s, \alpha, t) \wedge (\varrho(s) \vee f_\perp) \mid s \in C_1 \wedge t \in C_2 \,\}.$$

- $s'_* = [s_*]_{\simeq_{S'}}$.

Note that the feature $f_\perp$ is also included in the construction of new transition constraint function θ'. This prevents the feature $f_\perp$ from being stripped from the transition constraint function, which ensures that the coherent$^+$ quotients of a reversible FTS are still reversible. This will be discussed in more detail in the next section. The function that yields the set of coherent$^+$ feature bisimulation quotients of an FTS is denoted by $\leftrightarrows_{cf+}\text{-min}_{\texttt{FTS}}$.

We show that every FTS is feature bisimilar to its coherent$^+$ feature bisimulation quotients.

Theorem 7.5. *For each FTS fts and for all $fts' \in \leftrightarrows_{cf+}\text{-min}_{\texttt{FTS}}(fts)$ it holds that $fts' \leftrightarrows_f fts$.*

Proof. Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, and let $fts' = (S', \mathcal{A}, \theta', s'_*)$ be a coherent$^+$ feature bisimulation quotient of fts.

We define the relation R such that:

$$\begin{aligned} R = &\{(s, [\varphi \wedge ((\varrho(s) \wedge \varrho(t)) \vee f_\perp)]_{\sim_\mathcal{P}}, C) \in S \times \widehat{\mathbb{B}}(\mathcal{F}) \times S' \mid \exists t \in C \colon fts \models s \xleftrightarrow{\varphi}_f t\} \cup \\ &\{(C, [\varphi \wedge ((\varrho(s) \wedge \varrho(t)) \vee f_\perp)]_{\sim_\mathcal{P}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S \mid \exists t \in C \colon fts \models s \xleftrightarrow{\varphi}_f t\} \end{aligned}$$

We have to show that R is feature bisimulation relation on $fts \uplus fts'$ such that $(s_*, \widehat{\mathsf{true}}, s'_*) \in R$.

Since $\leftrightarrows_f$ is reflexive, we find that, for all $s \in S$, $(s, [\varrho(s) \vee f_\perp]_{\sim_\mathcal{P}}, C) \in R$, where $C \in S'$ is such that $s \in C$. Using that $s_* \in s'_*$ by Definition 7.7, it immediately follows that $(s_*, \widehat{\mathsf{true}}, s'_*) \in R$. Hence, it remains to show that R is a feature bisimulation relation.

From the definition of R it immediately follows that it is symmetric, and hence it remains to show that R satisfies the conditions from Definition 4.5.

Consider a tuple $(s, [\varphi \wedge ((\varrho(s) \wedge \varrho(t)) \vee f_\perp)]_{\sim_\mathcal{P}}, C) \in S \times \mathbb{B}(\mathcal{F}) \times S'$ such that $fts \models s \xleftrightarrow{\varphi}_f t$ and $t \in C$, for some $t \in S$. Let t be such.

We prove the transfer condition. Suppose that $s \xrightarrow{\alpha|\psi} s'$, for some $\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$, and for some $s' \in S$. We first show that this transition can be mimicked by state t for all products satisfying $\varphi \wedge \psi$. Since $fts \models s \overset{\varphi}{\leftrightarrow}_f t$, there exists a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F})$ such that $fts \models s' \overset{\varphi'}{\leftrightarrow}_f t'$, for all $(t', \varphi') \in \mathcal{T}$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t',\varphi') \in \mathcal{T}} \theta(t, \alpha, t') \wedge \varphi'.$$

Now we show that C can mimic all transitions from t, which we use to conclude that C can mimic the transition from s to s'. Using that $t \in C$, from Definition 7.7 it follows that $(\theta(t, \alpha, t') \wedge (\varrho(t) \vee f_\perp)) \Rightarrow_{\mathcal{P}} \theta'(C, \alpha, C_{t'})$, with $t' \in C_{t'}$, for all $(t', \varphi') \in \mathcal{T}$. Since $t' \in C_{t'}$ and $fts \models s' \overset{\varphi'}{\leftrightarrow}_f t'$, by construction of R we have $(s', [\varphi' \wedge ((\varrho(s') \wedge \varrho(t')) \vee f_\perp)]_{\sim_{\mathcal{P}}}, C_{t'}) \in R$, for all $(t', \varphi') \in \mathcal{T}$. Using that $\varrho(s) \wedge \psi \Rightarrow_{\mathcal{P}} \varrho(s')$ and $\varrho(t) \wedge \varphi' \Rightarrow_{\mathcal{P}} \varrho(t')$, for all $(t', \varphi') \in \mathcal{T}$, by definition of reachability, we find that

$$((\varrho(s) \wedge \varrho(t)) \vee f_\perp) \wedge \bigvee_{(t',\varphi') \in \mathcal{T}} \theta(t, \alpha, t') \wedge \varphi' \;\Rightarrow_{\mathcal{P}} \bigvee_{(t',\varphi') \in \mathcal{T}} \theta'(C, \alpha, C_{t'}) \wedge \varphi' \wedge ((\varrho(s') \wedge \varrho(t')) \vee f_\perp),$$

and hence the transfer condition for this tuple is satisfied.

We prove the transfer conditions for the remaining tuples of R in two steps, using a case distinction.

1. First consider a tuple $(C, [\varrho(s) \vee f_\perp]_{\sim_{\mathcal{P}}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \in C$.

   Suppose that $C \xrightarrow{\alpha|\psi} C'$, for some $\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. From Definition 7.7 it follows that $\psi = \bigvee\{\theta(u, \alpha, u') \wedge (\varrho(u) \vee f_\perp) \mid u \in C \wedge u' \in C'\}$. Take such a pair $(u, u') \in C \times C'$. Since both $s \in C$ and $u \in C$, by Definition 7.7 we know that $fts \models s \leftrightarrows_{cf+} u$. Hence we can find a set $\mathcal{S}_{(u,u')} \subseteq S \times \mathbb{B}(\mathcal{F})$ for each tuple $(u, u') \in C \times C'$ such that $fts \models s' \overset{\varphi'}{\leftrightarrow}_f u'$, for all $(s', \varphi') \in \mathcal{S}_{(u,u')}$, and such that

   $$((\varrho(s) \wedge \varrho(u)) \vee f_\perp) \wedge \theta(u, \alpha, u') \;\Rightarrow_{\mathcal{P}} \bigvee_{(s',\varphi') \in \mathcal{S}_{(u,u')}} \theta(s, \alpha, s') \wedge \varphi'.$$

   For each tuple $(u, u') \in C \times C'$ we have that $fts \models s' \overset{\varphi'}{\leftrightarrow}_f u'$, and hence by construction of R we have $(C', [\varphi' \wedge ((\varrho(s') \wedge \varrho(u')) \vee f_\perp)]_{\sim_{\mathcal{P}}}, s') \in R$, for all $(s', \varphi') \in \mathcal{S}_{(u,u')}$. Using that $\varrho(s) \wedge \theta(s, \alpha, s') \Rightarrow_{\mathcal{P}} \varrho(s')$ and $\varrho(u) \wedge \theta(u, \alpha, u') \Rightarrow_{\mathcal{P}} \varrho(u')$, we find that

   $$(\varrho(s) \vee f_\perp) \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(u,u') \in C \times C'} \bigvee_{(s',\varphi') \in \mathcal{S}_{(u,u')}} \theta(s, \alpha, s') \wedge \varphi' \wedge ((\varrho(s') \wedge \varrho(u')) \vee f_\perp),$$

   and hence the transfer condition for this tuple is satisfied.

2. Lastly consider a tuple $(C, [\varphi \wedge ((\varrho(s) \wedge \varrho(t)) \vee f_\bot)]_{\sim_\mathcal{P}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$, such that $fts \models s \overset{\varphi}{\leftrightarrows}_f t$ and $t \in C$, for some $t \in S$. Let t be such.

   Suppose that $C \xrightarrow{\alpha|\psi} C'$, for some $\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. By construction of R we have $(C, [\varrho(t) \vee f_\bot]_{\sim_\mathcal{P}}, t) \in R$. We have already shown that the transfer conditions are satisfied for such pairs in R, and hence we can find a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F})$ such that $(C', \hat{\varphi}'_t, t') \in R$, for all $(t', \varphi'_t) \in \mathcal{T}$, and such that

$$(\varrho(t) \vee f_\bot) \wedge \psi \Rightarrow_\mathcal{P} \bigvee_{(t', \varphi'_t) \in \mathcal{T}} \theta(t, \alpha, t') \wedge \varphi'_t.$$

   Now we show that s can mimic all transitions from t, which we use to conclude that s can mimic the transition from C to C'. Since $fts \models s \overset{\varphi}{\leftrightarrows}_f t$, we can find sets $\mathcal{S}_{t'} \subseteq S \times \mathbb{B}(\mathcal{F})$ for each $(t', \varphi'_t) \in \mathcal{T}$ such that, for all $(s', \varphi'_s) \in \mathcal{S}_{t'}$, we have $fts \models s' \overset{\varphi'_s}{\leftrightarrows}_f t'$ and such that

$$\varphi \wedge \theta(t, \alpha, t') \Rightarrow_\mathcal{P} \bigvee_{(s', \varphi'_s) \in \mathcal{S}_{t'}} \theta(s, \alpha, s') \wedge \varphi'_s.$$

   Since $(C', \hat{\varphi}'_t, t') \in R$, for all $(t', \varphi'_t) \in \mathcal{T}$, by definition of R this means that $\varphi'_t \sim_\mathcal{P} \varphi''_t \wedge ((\varrho(t') \wedge \varrho(u_{t'})) \vee f_\bot)$, for some $\varphi''_t \in \mathbb{B}(\mathcal{F})$ and some $u_{t'} \in C'$, such that $fts \models t' \overset{\varphi''_t}{\leftrightarrows}_f u_{t'}$. Since $fts \models s' \overset{\varphi'_s}{\leftrightarrows}_f t'$ for all $(s', \varphi'_s) \in \mathcal{S}_{t'}$, we derive $fts \models s' \overset{\varphi'_s \wedge \varphi''_t}{\leftrightarrows}_f u_{t'}$ using productwise transitivity. By construction of R we have $(C', [\varphi'_s \wedge \varphi''_t \wedge ((\varrho(s') \wedge \varrho(u_{t'})) \vee f_\bot)]_{\sim_\mathcal{P}}, s') \in R$. Using that $\varrho(s) \wedge \theta(s, \alpha, s') \Rightarrow_\mathcal{P} \varrho(s')$ and $\varphi'_t \Rightarrow_\mathcal{P} \varphi''_t \wedge (\varrho(u_{t'}) \vee f_\bot)$, we find that

$$\varphi \wedge \bigvee_{(t', \varphi'_t) \in \mathcal{T}} \theta(t, \alpha, t') \wedge \varphi'_t \Rightarrow_\mathcal{P} \bigvee_{(t', \varphi'_t) \in \mathcal{T}} \bigvee_{(s', \varphi'_s) \in \mathcal{S}_{t'}} \theta(s, \alpha, s') \wedge \varphi'_s \wedge \varphi'_t,$$

   and hence that

$$\varphi \wedge ((\varrho(s) \wedge \varrho(t)) \vee f_\bot) \wedge \psi \Rightarrow_\mathcal{P} \bigvee_{(t', \varphi'_t) \in \mathcal{T}} \bigvee_{(s', \varphi'_s) \in \mathcal{S}_{t'}} \theta(s, \alpha, s') \wedge \varphi'_s \wedge \varphi''_t \wedge ((\varrho(s') \wedge \varrho(u_{t'})) \vee f_\bot),$$

   from which we conclude that the transfer condition for this pair is also satisfied.

Hence we can conclude that $fts' \leftrightarrows_f fts$. □

With this theorem in place, in the next section we continue by proving that minimization of an FKS corresponds to minimization of its embedding$^+$ into an FTS.

7.1.4 Minimization using the embedding $\texttt{fts}^+$

Using the new embeddings and quotients defined in the previous sections, we will show that we can find the naive ϕ quotient of an FKS by finding the naive $\phi \vee f_\bot$-quotient of its corresponding FTS, for any $\phi \in \mathbb{B}(\mathcal{F})$, and that we can find the coherent quotients of an FKS by finding the coherent$^+$ quotients of its corresponding FTS.

We first show that, for any $\phi \in \mathbb{B}(\mathcal{F})$, the $\overset{\phi \vee f_\bot}{\leftrightarrows}_f$-quotient of a reversible FTS also is reversible.

Theorem 7.6. *Let fts be an arbitrary reversible FTS. Then* $\overset{\phi \vee f_\perp}{\longleftrightarrow}_f\text{-min}_{\mathtt{FTS}}(\mathit{fts})$ *is reversible, for all* $\phi \in \mathbb{B}(\mathcal{F})$.

Proof. Let $\mathit{fts} = (S, \mathcal{A}, \theta, s_*, \mathcal{F}, \mathcal{P})$ be a reversible FTS, and $\phi \in \mathbb{B}(\mathcal{F})$ be a feature expression. Let $\overset{\phi \vee f_\perp}{\longleftrightarrow}_f\text{-min}_{\mathtt{FTS}}(\mathit{fts}) = (S', \mathcal{A}, \theta', s'_*, \mathcal{F}, \mathcal{P})$. Let ϱ and ϱ' denote the reachability functions of *fts* and $\overset{\phi \vee f_\perp}{\longleftrightarrow}_f\text{-min}_{\mathtt{FTS}}(\mathit{fts})$, respectively. We show that $\overset{\phi \vee f_\perp}{\longleftrightarrow}_f\text{-min}_{\mathtt{FTS}}(\mathit{fts})$ is reversible.

1. $\mathcal{A} = 2^{AP} \cup \{\perp\}$ for some set AP follows directly from reversibility of *fts*.

2. $f_\perp \in \mathcal{F}$ and $\{f_\perp\} \in \mathcal{P}$ and $\forall P \in \mathcal{P}\colon (\, P \neq \{f_\perp\} \Rightarrow P \not\models f_\perp \,)$ follows directly from reversibility of *fts*.

3. Since *fts* is reversible, we know that $S = (S_\perp, \bar{S})$. We furthermore know that $s_\perp \xrightarrow{\perp|\psi}$ for some $\psi \in \mathbb{B}(\mathcal{F})$ such that $f_\perp \Rightarrow \psi$, for all $s_\perp \in S_\perp$, and that $\bar{s} \overset{\perp}{\not\to}$ for all $\bar{s} \in \bar{S}$. By definition of feature bisimulation we derive that $s \overset{f_\perp}{\longleftrightarrow}_f t$ implies that $s, t \in S_\perp$ or $s, t \in \bar{S}$, for all $s, t \in S$. By definition of $\overset{\phi \vee f_\perp}{\longleftrightarrow}_f\text{-min}_{\mathtt{FTS}}$ it follows that $s, t \in C$ implies $s, t \in S_\perp$ or $s, t \in \bar{S}$, for all $C \in S'$. Hence we can find a partition $(S'_\perp, \bar{S}')$ of S' where $S'_\perp = \{C \in S' \mid \forall C \subseteq S_\perp\}$ and $\bar{S}' = \{C \in S' \mid \forall C \subseteq \bar{S}\}$.

4. Since $s_* \in S_\perp$ and $s_* \in s'_*$, it follows that $s'_* \in S'_\perp$.

5. For all $C_1, C_2 \in S'_\perp$, $\bar{C}_1, \bar{C}_2 \in \bar{S}'$, and $\alpha, \alpha' \in \mathcal{A}_\tau$ we have

   (a) For all $s \in C_1$ and $t \in C_2$ we have $s, t \in S_\perp$ and hence $(f_\perp \wedge \theta(s, \alpha, t)) \sim_{\mathcal{P}} \mathsf{false}$. By definition of $\overset{\phi \vee f_\perp}{\longleftrightarrow}_f\text{-min}_{\mathtt{FTS}}$ it follows that $(f_\perp \wedge \theta'(C_1, \alpha, C_2)) \sim_{\mathcal{P}} \mathsf{false}$.

   (b) For all $s \in C_1$ and $t \in \bar{C}_1$ we have $s \in S_\perp$ and $t \in \bar{S}$ and hence $\theta(s, \perp, t) \not\sim_{\mathcal{P}} \mathsf{false}$ implies $f_\perp \Rightarrow_{\mathcal{P}} \theta(s, \perp, t)$. By definition of $\overset{\phi \vee f_\perp}{\longleftrightarrow}_f\text{-min}_{\mathtt{FTS}}$ it follows that $\theta'(C_1, \perp, \bar{C}_1) \not\sim_{\mathcal{P}} \mathsf{false}$ implies $f_\perp \Rightarrow_{\mathcal{P}} \theta'(C_1, \perp, \bar{C}_1)$.

   (c) For all $s \in C_1$ we have $s \in S_\perp$ and hence $(f_\perp \vee \varrho(s)) \Rightarrow_{\mathcal{P}} \bigvee_{\bar{s} \in \bar{S}} \theta(s, \perp, \bar{s})$. We have $\varrho'(C_1) \sim_{\mathcal{P}} \bigvee_{s \in C_1} \varrho(s)$, and hence we can derive

   $$f_\perp \vee \varrho'(C_1) \Rightarrow_{\mathcal{P}} \bigvee_{s \in C_1} \bigvee_{\bar{s} \in \bar{S}} \theta(s, \perp, \bar{s}).$$

   By definition of $\overset{\phi \vee f_\perp}{\longleftrightarrow}_f\text{-min}_{\mathtt{FTS}}$ we have

   $$\bigvee_{s \in C_1} \bigvee_{\bar{s} \in \bar{S}} \theta(s, \perp, \bar{s}) \sim_{\mathcal{P}} \bigvee_{\bar{C} \in \bar{S}'} \theta'(C_1, \perp, \bar{C}),$$

   and hence we have $f_\perp \vee \varrho'(C_1) \Rightarrow_{\mathcal{P}} \bigvee_{\bar{C} \in \bar{S}'} \theta'(C_1, \perp, \bar{C})$.

   (d) For all $s \in \bar{C}_1$ and $t \in \bar{C}_2$ we have $s, t \in \bar{S}$ and hence $\theta(s, \alpha, t) \sim_{\mathcal{P}} \mathsf{false}$. By definition of $\overset{\phi \vee f_\perp}{\longleftrightarrow}_f\text{-min}_{\mathtt{FTS}}$ it follows that $\theta'(\bar{C}_1, \alpha, \bar{C}_2) \sim_{\mathcal{P}} \mathsf{false}$.

(e) Assume $\theta'(\bar{C}_1, \alpha, C) \not\sim_{\mathcal{P}} \mathsf{false}$ and $\theta'(\bar{C}_1, \alpha', C') \not\sim_{\mathcal{P}} \mathsf{false}$, for some $C, C' \in S'_\perp$. By definition of $\xleftrightarrow{\phi \vee f_\perp}_f$-$\min_{\mathtt{FTS}}$ this implies there exist $s, t \in \bar{C}_1$ such that $\theta(s, \alpha, s') \not\sim_{\mathcal{P}} \mathsf{false}$ and $\theta(t, \alpha', t') \not\sim_{\mathcal{P}} \mathsf{false}$, for some $s' \in C$ and some $t' \in C'$, and hence $s', t' \in S_\perp$. By reversibility of *fts*, using condition 6(a) it follows that $\alpha = L(s')$, from which we derive $f_\perp \Rightarrow_{\mathcal{P}} \theta(s, \alpha, s')$ using conditions 6(b) and 5(b). By definition of $\xleftrightarrow{\phi \vee f_\perp}_f$-$\min_{\mathtt{FTS}}$ we know that $fts \models s \xleftrightarrow{f_\perp}_f t$, and hence t must be able to mimic the α-step for $f_\perp$. By conditions 5(d) and 5(e) we know that for all $\beta, \beta' \in \mathcal{A}_\tau$ it holds that $t \xrightarrow{\beta}$ and $t \xrightarrow{\beta'}$ implies $\beta = \beta'$, from which it follows that the transfer conditions can only be satisfied if $\alpha = \alpha'$, which satisfies this condition.

6. From conditions 5(c) and 6(b) it follows that each state $s_\perp \in S_\perp$ has at least one incoming transition with label $L(s_\perp)$.

   We show that for all $s, t \in C_\perp$, for each $C_\perp \in S'_\perp$, we have $L(s) = L(t)$. Using conditions 5(b), 5(c) and 6(b) we derive that $\theta(s, \perp, \bar{s}) \sim_{\mathcal{P}} \theta(\bar{s}, L(s), s)$ and $f_\perp \Rightarrow_{\mathcal{P}} \theta(s, \perp, \bar{s})$, for some $\bar{s} \in \bar{S}$. By definition of $\xleftrightarrow{\phi \vee f_\perp}_f$-$\min_{\mathtt{FTS}}$ we know that $fts \models s \xleftrightarrow{f_\perp}_f t$ and hence $f_\perp \Rightarrow_{\mathcal{P}} \theta(t, \perp, \bar{t})$, for some $\bar{t} \in \bar{S}$ such that $fts \models \bar{s} \xleftrightarrow{f_\perp}_f \bar{t}$. By conditions 5($d$) and 6($a$) it follows that $\theta(\bar{t}, \alpha, t') \not\sim_{\mathcal{P}} \mathsf{false}$ implies $\alpha = L(t)$, for all $t' \in S$, and hence it must be the case that $L(s) = L(t)$ for the transfer conditions to be satisfied.

   Now we can define $L(C_\perp) = L(s)$, for all $s \in C_\perp$, for all $C_\perp \in S'_\perp$.

   For all $C_\perp, C'_\perp \in S'_\perp$, $\bar{C} \in \bar{S}'$, and $\alpha \in \mathcal{A}_\tau$ we have:

   (a) For all $s \in \bar{C}$ we have $s \in \bar{S}$ and hence $\theta(s, \alpha, t) \not\sim_{\mathcal{P}} \mathsf{false}$ implies $\alpha = L(t)$, for all $t \in C_\perp$. By definition of $\xleftrightarrow{\phi \vee f_\perp}_f$-$\min_{\mathtt{FTS}}$, and since $L(C_\perp) = L(t)$, for all $t \in C_\perp$, it follows that $\theta'(\bar{C}, \alpha, C_\perp) \not\sim_{\mathcal{P}} \mathsf{false}$ implies $\alpha = L(C_\perp)$.

   (b) For each pair of states $(s, t) \in \bar{C} \times C_\perp$ we have $\theta(s, L(t), t) \sim_{\mathcal{P}} \theta(t, \perp, s)$. Hence, by definition of $\xleftrightarrow{\phi \vee f_\perp}_f$-$\min_{\mathtt{FTS}}$ it follows that

   $$\theta'(\bar{C}, L(C_\perp), C_\perp) = \bigvee_{(s,t) \in \bar{C} \times C_\perp} \theta(s, L(t), t) \sim_{\mathcal{P}} \bigvee_{(s,t) \in \bar{C} \times C_\perp} \theta(t, \perp, s) = \theta'(C_\perp, \perp, \bar{C}).$$

   (c) For each pair of states $(t, s) \in C'_\perp \times C_\perp$ we have $L(t) = L(s)$ and $\theta(t, \alpha, s) \not\sim_{\mathcal{P}} \mathsf{false}$ implies $\alpha = \tau$. By definition of $\xleftrightarrow{\phi \vee f_\perp}_f$-$\min_{\mathtt{FTS}}$ it follows that $L(C'_\perp) = L(C_\perp)$ and $\theta'(C'_\perp, \alpha, C_\perp) \not\sim_{\mathcal{P}} \mathsf{false}$ implies $\alpha = \tau$.

   (d) For each pair of states $(t, s) \in C'_\perp \times C_\perp$ we have $L(t) \neq L(s)$ and $\theta(t, \alpha, s) \not\sim_{\mathcal{P}} \mathsf{false}$ implies $\alpha = L(s)$. By definition of $\xleftrightarrow{\phi \vee f_\perp}_f$-$\min_{\mathtt{FTS}}$ it follows that $L(C'_\perp) \neq L(C_\perp)$ and $\theta'(C'_\perp, \alpha, C_\perp) \not\sim_{\mathcal{P}} \mathsf{false}$ implies $\alpha = L(s) = L(C_\perp)$.

We conclude that $\xleftrightarrow{\phi \vee f_\perp}_f$-$\min_{\mathtt{FTS}}(fts)$ is indeed reversible. □

We will need the following lemma to show that we can find the $\overset{\phi}{\underline{\leftrightarrow}}_f$-quotient of an FKS by finding the $\overset{\phi \vee f_\perp}{\underline{\longleftrightarrow}}_f$-quotient of its corresponding FTS.

Lemma 7.5. *We have* $\overset{\phi \vee f_\perp}{\underline{\longleftrightarrow}}_f\text{-min}_{\mathtt{FTS}} \circ \mathtt{fts}^+ = \mathtt{fts}^+ \circ \overset{\phi}{\underline{\leftrightarrow}}_f\text{-min}_{\mathtt{FKS}}$*, for all* $\phi \in \mathbb{B}(\mathcal{F})$.

Proof. Consider an FKS $fks = (S,\ AP,\ \theta,\ L,\ s_*,\ \mathcal{F},\ \mathcal{P})$. Let the FTS $fts_i = (S_i,\ \mathcal{A}_i,\ \theta_i,\ s_{*\,i},\ \mathcal{F}_i,\ \mathcal{P}_i)$ be such that $fts_1 = \overset{\phi \vee f_\perp}{\underline{\longleftrightarrow}}_f\text{-min}_{\mathtt{FTS}}(\mathtt{fts}^+(fks))$ and $fts_2 = \mathtt{fts}^+(\overset{\phi}{\underline{\leftrightarrow}}_f\text{-min}_{\mathtt{FKS}}(fks))$, for some $\phi \in \mathbb{B}(\mathcal{F})$.

We establish the isomorphism by proving that there are isomorphisms between S_1 and S_2, $\mathcal{A}_1$ and $\mathcal{A}_2$, θ_1 and θ_2, $s_{*\,1}$ and $s_{*\,2}$, $\mathcal{F}_1$ and $\mathcal{F}_2$, and $\mathcal{P}_1$ and $\mathcal{P}_2$.

- The set of states of $\overset{\phi}{\underline{\leftrightarrow}}_f\text{-min}_{\mathtt{FKS}}(fks)$ is, by definition of $\overset{\phi}{\underline{\leftrightarrow}}_f\text{-min}_{\mathtt{FKS}}$, $[S]^{\mathtt{FKS}}_{\overset{\phi}{\underline{\leftrightarrow}}_f}$.

  Hence we have $S_2 = [S]^{\mathtt{FKS}}_{\overset{\phi}{\underline{\leftrightarrow}}_f} \cup \{\bar{C} \mid C \in [S]^{\mathtt{FKS}}_{\overset{\phi}{\underline{\leftrightarrow}}_f}\}$.

  The set of states of $\mathtt{fts}^+(fks)$ is $S' = S \cup \bar{S}$, where $\bar{S} = \{\bar{s} \mid s \in S\}$. Since $s \xrightarrow{\perp | f_\perp}$, for all $s \in S$, and $\bar{s} \overset{\perp}{\not\longrightarrow}$, for all $\bar{s} \in \bar{S}$, we know that $\mathtt{fts}^+(fks) \not\models s \overset{\varphi \vee f_\perp}{\underline{\longleftrightarrow}}_f \bar{s}$, for all $(s, \bar{s}) \in S \times \bar{S}$. Hence we have $S_1 = [S]^{\mathtt{FTS}}_{\overset{\phi \vee f_\perp}{\underline{\longleftrightarrow}}_f} \cup [\bar{S}]^{\mathtt{FTS}}_{\overset{\phi \vee f_\perp}{\underline{\longleftrightarrow}}_f}$.

  In order to show that $S_1 = S_2$, it suffices to establish that there are isomorphisms between $[S]^{\mathtt{FKS}}_{\overset{\phi}{\underline{\leftrightarrow}}_f}$ and $[S]^{\mathtt{FTS}}_{\overset{\phi \vee f_\perp}{\underline{\longleftrightarrow}}_f}$, and between $[\bar{S}]^{\mathtt{FTS}}_{\overset{\phi \vee f_\perp}{\underline{\longleftrightarrow}}_f}$ and $\{\bar{C} \mid C \in [S]^{\mathtt{FKS}}_{\overset{\phi}{\underline{\leftrightarrow}}_f}\}$.

  Using Theorem 7.3 we immediately obtain $[S]^{\mathtt{FKS}}_{\overset{\phi}{\underline{\leftrightarrow}}_f} = [S]^{\mathtt{FTS}}_{\overset{\phi \vee f_\perp}{\underline{\longleftrightarrow}}_f}$.

  Furthermore, we can combine Theorem 7.3 with Lemma 7.2 to derive that $\mathtt{fts}^+(fks) \models \bar{s} \overset{\phi \vee f_\perp}{\underline{\longleftrightarrow}}_f \bar{t}$ iff $fks \models s \overset{\phi}{\underline{\leftrightarrow}}_f t$, for all $s, t \in S$, from which it follows that $[\bar{S}]^{\mathtt{FTS}}_{\overset{\phi \vee f_\perp}{\underline{\longleftrightarrow}}_f} = \{\bar{C} \mid C \in [S]^{\mathtt{FKS}}_{\overset{\phi}{\underline{\leftrightarrow}}_f}\}$.

- Since feature bisimulation reduction does not modify the set of actions, we immediately obtain $\mathcal{A}_1 = 2^{AP} \cup \{\perp\} = \mathcal{A}_2$. We will refer to $\mathcal{A}_1$ and $\mathcal{A}_2$ as $\mathcal{A}$.

- The transition constraint function of $\mathtt{fts}^+(fks)$ is $\theta' : S' \times \mathcal{A}_\tau \times S' \to \mathbb{B}(\mathcal{F}')$, which is defined such that, for all $s, t \in S'$ and $\alpha \in \mathcal{A}_\tau$:

$$\theta'(s,\alpha,t) = \begin{cases} \mathsf{true} & \text{if } s \in S \wedge t = \bar{s} \wedge \alpha = \perp \\ \mathsf{true} & \text{if } t \in S \wedge s = \bar{t} \wedge \alpha = L(t) \\ \theta(s,t) \wedge \neg f_\perp & \text{if } s,t \in S \wedge L(s) = L(t) \wedge \alpha = \tau \\ \theta(s,t) \wedge \neg f_\perp & \text{if } s,t \in S \wedge L(s) \neq L(t) \wedge \alpha = L(t) \\ \mathsf{false} & \text{otherwise} \end{cases}$$

  We have $\theta_1 : S_1 \times \mathcal{A}_\tau \times S_1 \to \mathbb{B}(\mathcal{F}_1)$ is such that, for all $C_1, C_2 \in S_1$ and $\alpha \in \mathcal{A}_\tau$:

$$\theta_1(C_1, \alpha, C_2) = \bigvee\{\, \theta'(s, \alpha, t) \mid s \in C_1 \wedge t \in C_2 \,\}.$$

The transition constraint function of $\overset{\phi}{\underset{f}{\leftrightarrow}}$-$\min_{\mathtt{FKS}}(\mathit{fks})$ is $\theta'' : [S]^{\mathtt{FKS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}} \times [S]^{\mathtt{FKS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}} \to \mathbb{B}(\mathcal{F})$, which is defined such that, for all $C_1, C_2 \in [S]^{\mathtt{FKS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}}$:

$$\theta''(C_1, C_2) = \bigvee\{\, \theta(s, t) \mid s \in C_1 \wedge t \in C_2 \,\}.$$

We have $\theta_2 : S_2 \times \mathcal{A}_\tau \times S_2 \to \mathbb{B}(\mathcal{F}_2)$ is such that, $C_1, C_2 \in S_2$ and $\alpha \in \mathcal{A}_\tau$:

$$\theta_2(C_1, \alpha, C_2) = \begin{cases} \mathsf{true} & \text{if } C_1 \in [S]^{\mathtt{FKS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}} \wedge C_2 = \bar{C}_1 \wedge \alpha = \bot \\ \mathsf{true} & \text{if } C_2 \in [S]^{\mathtt{FKS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}} \wedge C_1 = \bar{C}_2 \wedge \alpha = L(C_2) \\ \theta''(C_1, C_2) \wedge \neg f_\bot & \text{if } C_1, C_2 \in [S]^{\mathtt{FKS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}} \wedge L(C_1) = L(C_2) \wedge \alpha = \tau \\ \theta''(C_1, C_2) \wedge \neg f_\bot & \text{if } C_1, C_2 \in [S]^{\mathtt{FKS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}} \wedge L(C_1) \neq L(C_2) \wedge \alpha = L(t) \\ \mathsf{false} & \text{otherwise} \end{cases}$$

Here, $L(C) = L(s)$ for all $s \in C$, for all $C \in [S]^{\mathtt{FKS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}}$.

We will now show that $\theta_1 = \theta_2$. Pick two states $C_1, C_2 \in S_1$ and action $\alpha \in \mathcal{A}_\tau$. We distinguish four cases:

1. $C_1, C_2 \in [S]^{\mathtt{FTS}}_{\underset{f}{\overset{\phi \vee f_\bot}{\longleftrightarrow}}}$. In this case the images of C_1 and C_2 in S_2 are C_1 and C_2, respectively. Since $C_1, C_2 \in [S]^{\mathtt{FTS}}_{\underset{f}{\overset{\phi \vee f_\bot}{\longleftrightarrow}}}$ we have $s, t \in S$, for all $s \in C_1$ and $t \in C_2$.
   Now we distinguish two cases:
   (a) $\alpha = \tau$ and $L(C_1) = L(C_2)$, or $\alpha \neq \tau$ and $L(C_1) \neq L(C_2)$. We find

   $$\theta_1(C_1, \alpha, C_2) = \bigvee\{\, \theta(s, t) \wedge \neg f_\bot \mid s \in C_1 \wedge t \in C_2 \,\},$$

   and

   $$\theta_2(C_1, \alpha, C_2) = \bigvee\{\, \theta(s, t) \mid s \in C_1 \wedge t \in C_2 \,\} \wedge \neg f_\bot.$$

   (b) $\alpha = \tau$ and $L(C_1) \neq L(C_2)$, or $\alpha \neq \tau$ and $L(C_1) = L(C_2)$. We find

   $$\theta_1(C_1, \alpha, C_2) = \mathsf{false} = \theta_2(C_1, \alpha, C_2).$$

   In both cases we find that $\theta_1(C_1, \alpha, C_2) = \theta_2(C_1, \alpha, C_2)$.

2. $C_1 \in [S]^{\mathtt{FTS}}_{\overleftrightarrow{\phi \vee f_\perp}_f}$ and $C_2 \in [\bar{S}]^{\mathtt{FTS}}_{\overleftrightarrow{\phi \vee f_\perp}_f}$. Let C_3 denote the corresponding class of C_2 in $[S]^{\mathtt{FTS}}_{\overleftrightarrow{\phi \vee f_\perp}_f}$. That is, $C_3 = \{s \in S \colon \bar{s} \in C_2\}$.
   In this case the images of C_1 and C_2 in S_2 are C_1 and $\bar{C_3}$, respectively.
   We distinguish two cases:
   (a) $\alpha = \perp$ and $C_3 = C_1$, which implies $\bar{C_3} = \bar{C_1}$. We find
   $$\theta_1(C_1, \alpha, C_2) = \mathsf{true} = \theta_2(C_1, \alpha, \bar{C_3}).$$
   (b) $\alpha \neq \perp$ or $C_3 \neq C_1$. We find
   $$\theta_1(C_1, \alpha, C_2) = \mathsf{false} = \theta_2(C_1, \alpha, \bar{C_3}).$$
   In both cases we find that $\theta_1(C_1, \alpha, C_2) = \theta_2(C_1, \alpha, \bar{C_3})$.
3. $C_1 \in [\bar{S}]^{\mathtt{FTS}}_{\overleftrightarrow{\phi \vee f_\perp}_f}$ and $C_2 \in [S]^{\mathtt{FTS}}_{\overleftrightarrow{\phi \vee f_\perp}_f}$. Let C_3 denote the corresponding class of C_1 in $[S]^{\mathtt{FTS}}_{\overleftrightarrow{\phi \vee f_\perp}_f}$. That is, $C_3 = \{s \in S \colon \bar{s} \in C_1\}$.
   In this case the images of C_1 and C_2 in S_2 are $\bar{C_3}$ and C_2, respectively.
   We distinguish two cases:
   (a) $\alpha = L(C_2)$ and $C_3 = C_2$, which implies $\bar{C_3} = \bar{C_2}$. We find
   $$\theta_1(C_1, \alpha, C_2) = \mathsf{true} = \theta_2(\bar{C_3}, \alpha, C_2).$$
   (b) $\alpha \neq L(C_2)$ or $C_3 \neq C_2$. We find
   $$\theta_1(C_1, \alpha, C_2) = \mathsf{false} = \theta_2(\bar{C_3}, \alpha, C_2).$$
   In both cases we find that $\theta_1(C_1, \alpha, C_2) = \theta_2(\bar{C_3}, \alpha, C_2)$.
4. $C_1, C_2 \in [\bar{S}]^{\mathtt{FTS}}_{\overleftrightarrow{\phi \vee f_\perp}_f}$. Let C_3, C_4 denote the corresponding classes of C_1, C_2 in $[S]^{\mathtt{FTS}}_{\overleftrightarrow{\phi \vee f_\perp}_f}$, respectively. That is, $C_3 = \{s \in S \colon \bar{s} \in C_1\}$ and $C_4 = \{s \in S \colon \bar{s} \in C_2\}$.
   In this case the images of C_1 and C_2 in S_2 are $\bar{C_3}$ and $\bar{C_4}$, respectively.
   We find $\theta_1(C_1, \alpha, C_2) = \mathsf{false} = \theta_2(\bar{C_3}, \alpha, \bar{C_4})$.

We conclude that $\theta_1 = \theta_2$.

- The embedding $\mathtt{fts}^+$ does not change the initial state, and hence we have $s_{*1} = [s_*]_{\simeq_{S_1}}$, and $s_{*2} = [s_*]_{\simeq_{S_2}}$. Since $S_1 = S_2$, it follows that $s_{*1} = s_{*2}$.
- Since feature bisimulation reduction does not modify the set of features, we immediately obtain $\mathcal{F}_1 = \mathcal{F} \cup f_\perp = \mathcal{F}_2$.
- Since feature bisimulation reduction does not modify the set of products, we immediately obtain $\mathcal{P}_1 = \mathcal{P} \cup \{f_\perp\} = \mathcal{P}_2$.

It follows that $\xleftrightarrow{\phi \vee f_\perp}_f\text{-min}_{\texttt{FTS}} \circ \texttt{fts}^+ = \texttt{fts}^+ \circ \xleftrightarrow{\phi}_f\text{-min}_{\texttt{FKS}}$. □

Using the lemma, we straightforwardly obtain the desired result:

Theorem 7.7. *We have* $\xleftrightarrow{\phi}_f\text{-min}_{\texttt{FKS}} = \texttt{fts}^{+-1} \circ \xleftrightarrow{\phi \vee f_\perp}_f\text{-min}_{\texttt{FTS}} \circ \texttt{fts}^+$*, for all* $\phi \in \mathbb{B}(\mathcal{F})$.

Proof. Lemma 7.5 gives us

$$\texttt{fts}^{+-1} \circ \xleftrightarrow{\phi \vee f_\perp}_f\text{-min}_{\texttt{FTS}} \circ \texttt{fts}^+ = \texttt{fts}^{+-1} \circ \texttt{fts}^+ \circ \xleftrightarrow{\phi}_f\text{-min}_{\texttt{FKS}}.$$

Using Theorem 7.4 we obtain our desired conclusion:

$$\texttt{fts}^{+-1} \circ \xleftrightarrow{\phi \vee f_\perp}_f\text{-min}_{\texttt{FTS}} \circ \texttt{fts}^+ = \xleftrightarrow{\phi}_f\text{-min}_{\texttt{FKS}}.$$

□

Since FKS and FTS may have multiple coherent quotients, we cannot prove the above theorem for the function $\leftrightarrows_{cf}\text{-min}_{\texttt{FKS}}$. However, we can show that the set of coherent quotients of an FKS *fks* corresponds to the set of coherent$^+$ quotients of the FTS $\texttt{fts}^+(fks)$.

We first show that the coherent$^+$ quotients of a reversible FTS are also reversible. Since the proof is similar to that of Theorem 7.6, it is omitted here and included in Appendix E.

Theorem 7.8. *Let fts be an arbitrary reversible FTS. Then the FTS each* $fts' \in \leftrightarrows_{cf+}\text{-min}_{\texttt{FTS}}(fts)$ *is also reversible.*

With this theorem in place, we show that for each coherent quotient of an FKS *fks*, we can find a corresponding coherent$^+$ quotient of the FTS $\texttt{fts}^+(fks)$. That is, we are showing that the set of coherent quotients of *fks* corresponds to a subset of the the set of coherent$^+$ quotients of $\texttt{fts}^+(fks)$.

Lemma 7.6. *Let fks be an FKS. For each FKS* $fks' \in \leftrightarrows_{cf}\text{-min}_{\texttt{FKS}}(fks)$ *there exists an FTS* $fts \in \leftrightarrows_{cf+}\text{-min}_{\texttt{FTS}}(\texttt{fts}^+(fks))$ *such that* $fks' = \texttt{fts}^{+-1}(fts)$.

Proof. Let $fks = (S_1, AP, \theta_1, L, s_{*1}, \mathcal{F}_1, \mathcal{P}_1)$ be an FKS, and let $fks' = (S_1', AP, \theta_1', L', s_{*1}', \mathcal{F}_1, \mathcal{P}_1)$ be a coherent quotient of *fks*. Let ϱ_1 denote the reachability function of *fks*. Let $\texttt{fts}^+(fks) = (S_2, \mathcal{A}, \theta_2, s_{*2}, \mathcal{F}_2, \mathcal{P}_2)$ be the corresponding FTS of *fks*, where $S_2 = S_1 \cup \{\bar{u} \mid u \in S_1\}$. Let ϱ_2 denote the reachability function of $\texttt{fts}^+(fks)$.

By definition of $\leftrightarrows_{cf}\text{-min}_{\texttt{FKS}}$, we know that S_1' is a partition of S_1. Furthermore, we know that for each class $C \in S_1'$, for each pair of states $s, t \in C$ we have that $fks \models s \leftrightarrows_{cf} t$. By Theorem 7.3 and Lemma 7.2 we derive that $\texttt{fts}^+(fks) \models s \leftrightarrows_{cf+} t$ and $\texttt{fts}^+(fks) \models \bar{s} \leftrightarrows_{cf+} \bar{t}$.

From this it follows that $S_2' = S_1' \cup \bigcup_{C \in S_1'} \{\bar{s} \mid s \in C\}$ is the state space of a coherent$^+$ quotient of $\texttt{fts}^+(fks)$. Let $fts' = (S_2', \mathcal{A}, \theta_2', s_{*2}', \mathcal{F}_2, \mathcal{P}_2)$ denote this quotient, and let $\texttt{fts}^{+-1}(fts') = (S_3, AP', \theta_3, L'', s_{*3}, \mathcal{F}_3, \mathcal{P}_3)$ be its corresponding FKS. We show that $fks' = \texttt{fts}^{+-1}(fts')$.

We establish the isomorphism by proving that there are isomorphism between S_1' and S_3, AP and AP', θ_1' and θ_3, L' and L'', s_{*1}' and s_{*3}, $\mathcal{F}_1$ and $\mathcal{F}_3$, and $\mathcal{P}_1$ and $\mathcal{P}_3$.

- We find that the set of states of $\texttt{fts}^{+\,-1}(\mathit{fts}')$ is

$$S_3 = \{s \in S_2' \mid s \xrightarrow{\perp}\} = \{s \in (\, S_1' \;\cup \bigcup_{C \in S_1'} \{\bar{s} \mid s \in C\}\,) \mid s \xrightarrow{\perp}\}.$$

  By definition of $\texttt{fts}^{+}$ and $\leftrightarrows_{cf}\text{-min}_{\texttt{FKS}}$ we know that this is equal to S_1'.

- The set of actions is unchanged by $\leftrightarrows_{cf+}\text{-min}_{\texttt{FTS}}$, as is the set of atomic propositions by $\leftrightarrows_{cf}\text{-min}_{\texttt{FKS}}$, and hence $AP = AP'$ follows from $\texttt{fts}^{+\,-1} \circ \texttt{fts}^{+} = \texttt{Id}$.

- Pick states $C_1, C_2 \in S_1'$. Note that the images of C_1 and C_2 in S_3 are C_1 and C_3, respectively.

  By definition of $\leftrightarrows_{cf}\text{-min}_{\texttt{FKS}}$ we have:

$$\theta_1'(C_1, C_2) = \bigvee\{\, \theta_1(s,t) \wedge \varrho_1(s) \mid s \in C_1 \wedge t \in C_2 \,\}.$$

  Pick states $(s,t) \in C_1 \times C_2$. By definition of $\texttt{fts}^{+}$ we have:

$$\theta_2(s,\alpha,t) = \begin{cases} \theta_1(s,t) \wedge \neg f_\perp & \text{if } L(s) = L(t) \wedge \alpha = \tau \\ \theta_1(s,t) \wedge \neg f_\perp & \text{if } L(s) \neq L(t) \wedge \alpha = L(t) \\ \mathsf{false} & \text{otherwise} \end{cases}$$

  By definition of $\leftrightarrows_{cf+}\text{-min}_{\texttt{FTS}}$ we have, for all $\alpha \in \mathcal{A}_\tau$:

$$\theta_2'(C_1, \alpha, C_2) = \bigvee\{\, \theta_2(s,\alpha,t) \wedge (\varrho_2(s) \vee f_\perp) \mid s \in C_1 \wedge t \in C_2 \,\},$$

  which is equal to:

$$\theta_2'(C_1, \alpha, C_2) = \begin{cases} \bigvee\{\theta(s,t) \wedge \neg f_\perp \wedge \varrho_2(s) \mid s,t \in C_1, C_2\} & \text{if } L'(C_1) = L'(C_2) \wedge \alpha = \tau \\ \bigvee\{\theta(s,t) \wedge \neg f_\perp \wedge \varrho_2(s) \mid s,t \in C_1, C_2\} & \text{if } L'(C_1) \neq L'(C_2) \wedge \alpha = L'(C_2) \\ \mathsf{false} & \text{otherwise} \end{cases}$$

  By definition of $\texttt{fts}^{+\,-1}$ we have:

$$\theta_3(C_1, C_2) = \theta_2'(C_1, L''(C_2), C_2) \vee \theta_2'(C_1, \tau, C_2).$$

  Using that $L' = L''$, this is equal to:

$$\theta_3(C_1, C_2) = \bigvee\{\theta(s,t) \wedge \neg f_\perp \wedge \varrho_2(s) \mid s,t \in C_1, C_2\}.$$

  Using that $\varrho_2(s) \wedge \neg f_\perp \sim_{\mathcal{P}_1} \varrho_1(s)$, for all $s \in S_1$, we conclude that $\theta_1' = \theta_3$.

- For all $C_2 \in S_3$, $L''(C_2) = a$ for the unique $a \in 2^{AP}$ such that $C_1 \xrightarrow{a} C_2$ in fts', for some $C_1 \in S_2'$. By construction of θ_2' we know we can pick $C_1 = \{\bar{s} \mid s \in C_2\}$ and $a = L'(C_2)$. Hence $L'' = L'$.

- The initial state is unchanged by the embedding $\texttt{fts}^{+-1}$, and hence we find $s_{*3} = s'_{*2}$, which is the unique class $C \in S'_2$ such that $s_{*2} \in C$. Since $s_{*2} = s_{*1}$ and $s_{*1} \in s'_{*1}$, we derive that $s_{*3} = s'_{*1}$.
- The set of features is unchanged by $\leftrightarrows_{cf}\text{-min}_{\texttt{FKS}}$, and hence $\mathcal{F}_1 = \mathcal{F}_3$ follows from $\texttt{fts}^{+-1} \circ \texttt{fts}^{+} = \texttt{Id}$.
- The set of products is unchanged by $\leftrightarrows_{cf}\text{-min}_{\texttt{FKS}}$, and hence $\mathcal{P}_1 = \mathcal{P}_3$ follows from $\texttt{fts}^{+-1} \circ \texttt{fts}^{+} = \texttt{Id}$.

We conclude that the set of coherent quotients of *fks* indeed corresponds to a subset of the set of coherent$^+$ quotients of $\texttt{fts}^+(\mathit{fks})$. □

Conversely, given an FKS *fks*, we show that for each coherent$^+$ quotient of the FTS $\texttt{fts}^+(\mathit{fks})$, we can find a corresponding coherent quotient of the *fks*. That is, we show that the set of coherent$^+$ quotients of $\texttt{fts}^+(\mathit{fks})$ corresponds to a subset of the set of coherent quotients of *fks*.

Lemma 7.7. *Let fks be an FKS. For each FTS fts* $\in \leftrightarrows_{cf+}\text{-min}_{\texttt{FTS}}(\texttt{fts}^+(\mathit{fks}))$ *there exists an FKS* $\mathit{fks}' \in \leftrightarrows_{cf}\text{-min}_{\texttt{FKS}}(\mathit{fks})$ *such that* $\mathit{fks}' = \texttt{fts}^{+-1}(\mathit{fts})$.

Proof. The proof is similar to that of Lemma 7.6. The interested reader is referred to Appendix E. □

Combining these two lemmas we find that the set of coherent quotients of an FKS *fks* corresponds exactly to the set of coherent$^+$ quotients of the FTS $\texttt{fts}^+(\mathit{fks})$.

Theorem 7.9. *Let fks be an FKS. Then*

$$\leftrightarrows_{cf}\text{-min}_{\texttt{FKS}}(\mathit{fks}) = \{\texttt{fts}^{-1}(\mathit{fts}) \mid \mathit{fts} \in \leftrightarrows_{cf+}\text{-min}_{\texttt{FTS}}(\texttt{fts}(\mathit{fks}))\}.$$

Proof. This follows immediately from Lemmas 7.6 and 7.7. □

Hence, in this section we have shown that it is possible to perform both naive and coherent feature bisimulation reduction on an FKS by performing this reduction on its embedding into an FTS.

We will now summarize the results obtained so far in this chapter. We started in Section 7.1.1 by proposing a straightforward generalization of the embedding `lts` to the product-family level, resulting in the embedding `fts`. We found that this embedding does preserve feature bisimulation, which was proved by Theorem 7.2. However, it did not preserve minimality modulo coherent feature bisimulation.

Therefore, we proposed a slightly adapted embedding $\texttt{fts}^+$ in Section 7.1.2, that uses a dummy feature and a dummy product to ensure that minimality is preserved. We found that also this embedding preserves minimality modulo feature bisimulation (Theorem 7.3), and furthermore we defined a reverse embedding $\texttt{fts}^{+-1}$ that is the left inverse of $\texttt{fts}^+$, as proven by Theorem 7.4. In order to make use of the dummy product and ensure preservation of minimality modulo coherent feature bisimulation, a slightly adapted definition of the coherent quotients was needed as well, and was proposed in Section 7.1.3. We proved that these coherent$^+$ quotients of an FTS *fts* are feature bisimilar to *fts* in Theorem 7.5, similar to the original coherent quotients.

Finally, in Section 7.1.4 we proved that we can find the naive feature bisimulation quotient of an FKS by performing naive feature bisimulation reduction to its corresponding FTS (Theorem 7.7), and that the set of coherent feature bisimulation quotients of an FKS exactly corresponds to the set of coherent$^+$ feature bisimulation quotients of its corresponding FTS (Theorem 7.9).

In the next section we will discuss minimization in event-based models through minimization in state-based models.

7.2 Minimization for event-based models

In this section we discuss the possibilities of minimizing event-based models by performing minimization for state-based models. Section 7.2.1 presents an embedding from event-based to state-based models, and Section 7.2.2 presents the possibilities for minimization using this embedding.

Many of the proofs in this section are similar to the proofs in Section 7.1, and are therefore omitted. The interested reader is referred to Appendix E.

7.2.1 Embeddings to state-based models

In [14] an embedding from LTS to KS has been proposed. In [26], a slight adaptation of this embedding is presented, and that preserves minimality modulo bisimulation. The definition of this adapted embedding is presented below.

Definition 7.8. *Let* $lts = (S, \mathcal{A}, \rightarrow_{lts}, s_*)$ *be an LTS. The embedding* $\texttt{ks}\colon \texttt{LTS} \rightarrow \texttt{KS}$ *is defined such that* $\texttt{ks}(lts) = (S', AP, \rightarrow_{ks}, L, s_*)$ *is a KS such that:*

- $S' = S \cup \{\bar{t}_a \mid t \in S \wedge a \in \mathcal{A} \wedge \exists s \in S\colon s \xrightarrow{a} t\}$,
- $AP = \mathcal{A} \cup \{\bot\}$, *where* $\bot \notin \mathcal{A}$,
- $\rightarrow_{ks} \subseteq S' \times S'$ *is the least relation satisfying the following rules for all* $s, t \in S$ *and* $a \in \mathcal{A}$*:*

$$\frac{s \xrightarrow{a}_{lts} t}{s \rightarrow_{ks} \bar{t}_a} \qquad \frac{}{\bar{t}_a \rightarrow_{ks} t} \qquad \frac{s \xrightarrow{\tau}_{lts} t}{s \rightarrow_{ks} t}$$

- $L(s) = \{\bot\}$ *for* $s \in S$, *and* $L(\bar{t}_a) = \{a\}$.

In this embedding a new state is introduced for each incoming action different from τ to an existing state, to facilitate the encoding of the transition labeling of the LTS. The fresh atomic proposition $\bot$ is used to label the states from the LTS. The newly introduced states are labeled with a singleton set containing the action it represents. An example of the application of the embedding `ks` is shown in Figure 7.7.

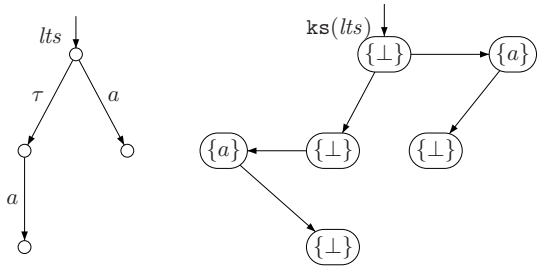


Figure 7.7: An LTS *lts* and its corresponding KS $\texttt{ks}(lts)$.

In [26] it has been proven that the embedding `ks` preserves and reflects bisimulation.

Theorem 7.10. *Let $lts = (S, \mathcal{A}, \rightarrow, s_*)$ be an LTS. Then, for all $s, s' \in S$, $lts \models s \leftrightarrows s'$ if and only if $\mathtt{ks}(lts) \models s \leftrightarrows s'$.*

Proof. As in Theorem 5.4 of [26]. □

The next is step is to find an embedding from event-based models to state-based models on the product family level. We propose a straightforward generalization from the embedding `ks` to an embedding `fks`, that transforms an FTS into an FKS.

Definition 7.9. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS. The embedding $\mathtt{fks}\colon \mathtt{FTS} \rightarrow \mathtt{FKS}$ is defined such that $\mathtt{fks}(fts) = (S', AP, \theta', L, s_*)$ is an FKS such that:*

- $S' = S \cup \{\bar{t}_a \mid t \in S \wedge a \in \mathcal{A} \wedge \exists s \in S\colon s \xrightarrow{a} t\}$,
- $AP = \mathcal{A} \cup \{\bot\}$, *where* $\bot \notin \mathcal{A}$,
- $\theta' : S' \times S' \rightarrow \mathbb{B}(\mathcal{F})$ *is constructed such that, for all* $s, s' \in S'$, $t \in S$ *and* $a \in \mathcal{A}$:

$$\theta'(s,s') = \begin{cases} \mathsf{true} & \text{if } s = \bar{t}_a \text{ and } s' = t \\ \theta(s,a,t) & \text{if } s \in S \text{ and } s' = \bar{t}_a \\ \theta(s,\tau,s') & \text{if } s, s' \in S \\ \mathsf{false} & \text{otherwise} \end{cases}$$

- $L(s) = \{\bot\}$ *for* $s \in S$, *and* $L(\bar{t}_a) = \{a\}$.

The embedding `fks` is identical to the embedding `ks`, apart from the feature expressions attached to the transitions. The outgoing transitions of the states that facilitate the encoding of the action information of the FTS are assigned the feature expression `true`. All other feature expressions are carried over from the FTS to the FKS. An example of the application of the embedding `fks` is shown in Figure 7.8.

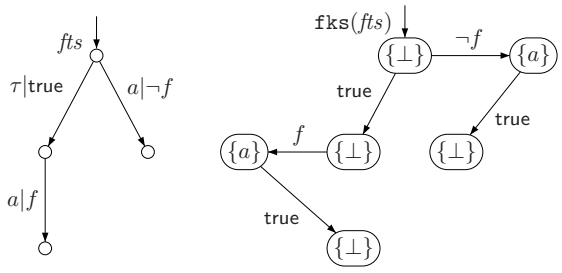


Figure 7.8: An FTS *fts* and its corresponding FKS $\mathtt{fks}(fts)$.

With Lemma 7.1 we proved that projection on a product distributes over the embedding from an event-based to a state-based model. This lemma was useful to lift properties of the embedding on the single-product level to the embedding on the product-family level. Unfortunately, for the embedding `fks` projection on a single product does not distribute over the embedding from a state-based to an event-based model. This is the case since the embeddings from state-based to event-based models introduce a new state for each transition. However, the projection operation removes transitions from the model, while it does not remove states. Hence we have a situation where applying the embedding before performing projection will result in a system with more states than the other way around. A simple example of this behavior is shown in Figure 7.9.

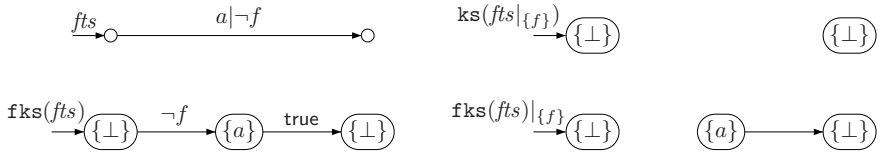


Figure 7.9: An FTS *fts*, for which the KS $\texttt{ks}(\mathit{fts}|_{\{f\}})$ is different from the KS $\texttt{fks}(\mathit{fts})|_{\{f\}}$.

Therefore, we will prove a slightly weaker lemma, stating that bisimulation is preserved when changing the order of the embedding an projection operation. The proof for this lemma can be found in Appendix E.

Lemma 7.8. *Let* $\mathit{fts} = (S, \mathcal{A}, \theta, s_*)$ *be an FTS. Then, for all* $P \in \mathcal{P}$ *and for all* $s, t \in s$ *we have* $\texttt{fks}(\mathit{fts})|_P \models s \leftrightarrows t \Leftrightarrow \texttt{ks}(\mathit{fts}|_P) \models s \leftrightarrows t$.

Using this lemma, we show that the embedding `fks` preserves and reflects feature bisimilarity.

Theorem 7.11. *Let* $\mathit{fts} = (S, \mathcal{A}, \theta, s_*)$ *be an FTS.Then, for all* $s, s' \in S$, *and* $\varphi \in \mathbb{B}(\mathcal{F})$, *we have* $\mathit{fts} \models s \overset{\varphi}{\leftrightarrows}_f s'$ *if and only if* $\texttt{fks}(\mathit{fts}) \models s \overset{\varphi}{\leftrightarrows}_f s'$.

We define a reverse embedding $\texttt{fks}^{-1}$ for the embedding `fks`. In order for this embedding to be well-defined, we first define which FKS are reversible.

Definition 7.10. *Let* $\mathit{fks} = (S, AP, \theta, L, s_*)$ *be an FKS. Then fks is reversible iff the following conditions are satisfied:*

1. $AP = \mathcal{A} \cup \{\bot\}$, *for some set* $\mathcal{A}$.
2. $|L(s)| = 1$ *for all* $s \in S$.
3. *For all* $s \in S$ *for which* $\bot \notin L(s)$, *we require that, for all* $s', s'' \in S$, $s \xrightarrow{\psi} s'$ *and* $s \xrightarrow{\psi'} s''$ *implies both* $s' = s''$ *and* $L(s') = \{\bot\}$ *and* $\psi = \psi' = \mathsf{true}$. *Furthermore, we require there exists a state* $s' \in S$ *such that* $s \xrightarrow{\mathsf{true}} s'$. *We will refer to this state* s' *as the target of* s.

Note that any embedding $\texttt{fks}(\mathit{fts})$ of an FTS *fks* is a reversible FKS. We now give the definition of the reverse embedding $\texttt{fks}^{-1}$.

Definition 7.11. *Let $fks = (S, AP, \theta, L, s_*)$ be a reversible FKS. The embedding $\texttt{fks}^{-1}\colon \texttt{FKS} \to \texttt{FTS}$ is defined such that $\texttt{fks}^{-1}(fks) = (S', \mathcal{A}, \theta', s_*)$ is an FTS such that:*

- $S' = \{s \in S \mid L(s) = \{\bot\}\}$,
- $\mathcal{A} = AP \setminus \{\bot\}$.
- $\theta'\colon S' \times \mathcal{A}_\tau \times S' \to \mathbb{B}(\mathcal{F})$ *is constructed such that, for all $s, t \in S'$, and $\alpha \in \mathcal{A}_\tau$:*

$$\theta'(s,\alpha,t) = \begin{cases} \theta(s,t) & \text{if } \alpha = \tau \wedge L(s) = L(t) \\ \bigvee\{\theta(s,s') \wedge \theta(s',t) \mid s' \in S \wedge \alpha \in L(s')\} & \text{if } \alpha \neq \tau \\ \mathsf{false} & \text{otherwise} \end{cases}$$

We show that the defined reverse embedding $\texttt{fks}^{-1}$ is the left inverse of the embedding $\texttt{fks}$, by proving that composing these embeddings results in the identity function.

Theorem 7.12. *We have $\texttt{fks}^{-1} \circ \texttt{fks} = \texttt{Id}$.*

Proof. Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS. Let $\texttt{fks}(fts) = (S', AP, \theta', L, s'_*)$ and $\texttt{fks}^{-1}(\texttt{fks}(fts)) = (S'', \mathcal{A}', \theta'', s''_*)$. We establish the isomorphism by proving that there are isomorphisms between S and S'', $\mathcal{A}$ and $\mathcal{A}'$, θ and θ'', and s_* and s''_*.

From the definition of $\texttt{fts}$ (applied to fks) it follows that

- $S' = S \cup \{\bar{t}_a \mid t \in S \wedge a \in \mathcal{A} \wedge \exists s \in S\colon s \xrightarrow{a} t\}$,
- $AP = \mathcal{A} \cup \{\bot\}$,
- For $s, s' \in S'$, $t \in S$ and $a \in \mathcal{A}$ we have:

$$\theta'(s,s') = \begin{cases} \mathsf{true} & \text{if } s = \bar{t}_a \text{ and } s' = t \\ \theta(s,a,t) & \text{if } s \in S \text{ and } s' = \bar{t}_a \\ \theta(s,\tau,s') & \text{if } s, s' \in S \\ \mathsf{false} & \text{otherwise} \end{cases}$$

- $L(s) = \{\bot\}$ for $s \in S$, and $L(\bar{t}_a) = \{a\}$.
- $s'_* = s_*$.

An application of $\texttt{fks}^{-1}$, applied to $\texttt{fks}(fts)$, gives:

- $S'' = \{s \in S' \mid L(s) = \{\bot\}\}$. Since $L(s) = \{\bot\}$ if and only if $s \in S$, we immediately obtain $S = S''$.
- $\mathcal{A}' = AP \setminus \{\bot\} = (\mathcal{A} \cup \{\bot\}) \setminus \{\bot\} = \mathcal{A}$.
- For $s, t \in S''$ and $\alpha \in \mathcal{A}_\tau$ we have:

$$\theta''(s,\alpha,t) = \begin{cases} \theta'(s,t) & \text{if } \alpha = \tau \wedge L(s) = L(s') \\ \bigvee\{\theta'(s,s') \wedge \theta'(s',t) \mid s' \in S \wedge \alpha \in L(s')\} & \text{if } \alpha \neq \tau \\ \mathsf{false} & \text{otherwise} \end{cases}$$

which is equivalent to:

$$\theta''(s,\alpha,t) = \begin{cases} \theta(s,\tau,t) & \text{if } \alpha = \tau \\ \bigvee\{\theta(s,a,t) \wedge \mathsf{true}\} & \text{if } \alpha \neq \tau \\ \mathsf{false} & \text{otherwise} \end{cases}$$

Since $S = S''$ the images of s and t in S are s and t, respectively, and hence we conclude $\theta'' = \theta$.

- $s''_* = s'_* = s_*$.

We conclude that $\mathtt{fks}^{-1} \circ \mathtt{fks} = \mathtt{Id}$. □

Hence we have shown that we can transform an FTS *fts* to and from a state-based model using the embedding `fks` and $\mathtt{fks}^{-1}$ without changing *fts*. In the next section we discuss how minimization of an FTS corresponds to minimization of its embedding into an FKS.

7.2.2 Naive minimization using the embedding `fks`

Using the embeddings defined in the previous section, we will show that we can find the naive quotient of an FTS by finding the naive quotient of its corresponding FKS.

We first show that the naive quotient of a reversible FKS also is reversible.

Lemma 7.9. *Let fks be an arbitrary reversible FKS. Then* $\overset{\phi}{\leftrightarrows}_f\text{-min}_{\mathtt{FKS}}(\mathit{fks})$ *is reversible, for all* $\phi \in \mathbb{B}(\mathcal{F})$.

Proof. Let $\mathit{fks} = (S,\ AP,\ \theta,\ L,\ s_*)$ be a reversible FKS, and let $\phi \in \mathbb{B}(\mathcal{F})$ be a feature expression. Let $\overset{\phi}{\leftrightarrows}_f\text{-min}_{\mathtt{FKS}}(\mathit{fks}) = (S',\ AP',\ \theta',\ L',\ s'_*)$. We show that $\overset{\phi}{\leftrightarrows}_f\text{-min}_{\mathtt{FKS}}(\mathit{fks})$ is reversible.

1. Since $AP' = AP$, $AP' = \mathcal{A} \cup \{\bot\}$ for some set $\mathcal{A}$ follows directly from $AP = \mathcal{A} \cup \{\bot\}$ for some set $\mathcal{A}$.

2. Since $|L(s)| = 1$ for all $s \in S$, it follows from the definition of $\overset{\phi}{\leftrightarrows}_f\text{-min}_{\mathtt{FKS}}$ that $|L'(s')| = 1$ for all $s' \in S'$.

3. Since *fks* is reversible we have that for all $s \in S$ for which $\bot \notin L(s)$, and for all $s', s'' \in S$, $s \xrightarrow{\psi} s'$ and $s \xrightarrow{\psi'} s''$ implies $s' = s''$ and $L(s') = \{\bot\}$ and $\psi = \psi' = \mathsf{true}$.

   Let $C \in S'$ be such that $\bot \notin L'(C)$. By reversibility of *fks* and definition of strong feature bisimilarity for FKS we find that for all $s, t \in C$ and for all $s', t' \in S$, $s \xrightarrow{\psi} s'$ and $t \xrightarrow{\psi'} t'$ implies $\mathit{fks} \models s' \overset{\phi}{\leftrightarrows}_f t'$. From this and the definition of $\overset{\phi}{\leftrightarrows}_f\text{-min}_{\mathtt{FKS}}$ it follows that for all $C', C'' \in S'$, $C \xrightarrow{\psi} C'$ and $C \xrightarrow{\psi'} C''$ implies $C' = C''$ and $L'(C') = \{\bot\}$ and $\psi = \psi' = \mathsf{true}$.

   Lastly, pick a state $s \in S$. Since *fks* is reversible there exists a state $s' \in S$ such that $s \xrightarrow{\mathsf{true}} s'$. Let $C' \in S'$ be such that $s' \in C'$. It follows that $C \xrightarrow{\mathsf{true}} C'$.

We conclude that $\overset{\phi}{\leftrightarrows}_f\text{-min}_{\mathtt{FKS}}(\mathit{fks})$ is indeed reversible. □

We prove the following lemma for reversible FKS, stating that two states that have the same label, and are not labeled with $\{\bot\}$, are feature bisimilar if and only if their target states are bisimilar. The proof can be found in Appendix E.

Lemma 7.10. *Let the FKS $fks = (S, AP, \theta, L, s_*)$ be a reversible FKS. For states $s, t \in S$ such that $L(s) \neq \{\bot\}$ and $L(t) \neq \{\bot\}$, and states $s', t' \in S$ such that $s \to s'$ and $t \to t'$ we have, for all $\varphi \in \mathbb{B}(\mathcal{F})$:*

$$fks \models s \overset{\varphi}{\leftrightarrows}_f t \Leftrightarrow (fks \models s' \overset{\varphi}{\leftrightarrows}_f t' \wedge L(s) = L(t)).$$

We will need the following lemma to show that we can find the naive quotient of an FTS by finding the naive quotient of its corresponding FKS.

Lemma 7.11. *We have $\overset{\phi}{\leftrightarrows}_f\text{-min}_{\texttt{FKS}} \circ \texttt{fks} = \texttt{fks} \circ \overset{\phi}{\leftrightarrows}_f\text{-min}_{\texttt{FTS}}$, for all $\phi \in \mathbb{B}(\mathcal{F})$.*

Proof. Consider an FTS $fts = (S, \mathcal{A}, \theta, s_*)$. Let the FKS $fks_i = (S_i, AP_i, \theta_i, L_i, s_{*i})$ be such that $fks_1 = \overset{\phi}{\leftrightarrows}_f\text{-min}_{\texttt{FKS}}(\texttt{fks}(fts))$ and $fks_2 = \texttt{fks}(\overset{\phi}{\leftrightarrows}_f\text{-min}_{\texttt{FTS}}(fts))$, for some $\phi \in \mathbb{B}(\mathcal{F})$.

We establish the isomorphism by proving that there are isomorphisms between S_1 and S_2, AP_1 and AP_2, θ_1 and θ_2, L_1 and L_2, and s_{*1} and s_{*2}.

- The set of states of $\overset{\phi}{\leftrightarrows}_f\text{-min}_{\texttt{FTS}}(fts)$ is, by definition of $\overset{\phi}{\leftrightarrows}_f\text{-min}_{\texttt{FTS}}$, $[S]^{\texttt{FTS}}_{\overset{\phi}{\leftrightarrows}_f}$.

  Hence we have $S_2 = [S]^{\texttt{FTS}}_{\overset{\phi}{\leftrightarrows}_f} \cup \overline{[S]^{\texttt{FTS}}_{\overset{\phi}{\leftrightarrows}_f}}$, where $\overline{[S]^{\texttt{FTS}}_{\overset{\phi}{\leftrightarrows}_f}} = \bigcup_{a \in \mathcal{A}} \overline{[S]^{\texttt{FTS}}_{\overset{\phi}{\leftrightarrows}_f}}|_a$, and where $\overline{[S]^{\texttt{FTS}}_{\overset{\phi}{\leftrightarrows}_f}}|_a = \{\bar{C}_a \mid C \in [S]^{\texttt{FTS}}_{\overset{\phi}{\leftrightarrows}_f} \wedge \exists C' \in [S]^{\texttt{FTS}}_{\overset{\phi}{\leftrightarrows}_f} : C' \xrightarrow{a} C\}$, for all $a \in \mathcal{A}$.

  The set of states of $\texttt{fks}(fts)$ is $S' = S \cup \bar{S}$, where $\bar{S} = \bigcup_{a \in \mathcal{A}} \bar{S}_a$, with $\bar{S}_a = \{\bar{t}_a \mid t \in S \wedge \exists s \in S : s \xrightarrow{a} t\}$, for all $a \in \mathcal{A}$. Furthermore, $L(s) = \{\bot\}$ for $s \in S$, and $L(\bar{t}_a) = \{a\}$, for $\bar{t}_a \in \bar{S}$. Hence, we know that $\texttt{fks}(fts) \not\models s \overset{\varphi}{\leftrightarrows}_f \bar{t}_a$, for all $(s, \bar{t}_a) \in S \times \bar{S}$, and for all $\varphi \in \mathbb{B}(\mathcal{F})$. Furthermore $\texttt{fks}(fts) \not\models \bar{s}_a \overset{\varphi}{\leftrightarrows}_f \bar{t}_b$ if $a \neq b$, for all $(\bar{s}_a, \bar{t}_b) \in \bar{S}_a \times \bar{S}_b$, and for all $\varphi \in \mathbb{B}(\mathcal{F})$.

  Hence we have $S_1 = [S]^{\texttt{FKS}}_{\overset{\phi}{\leftrightarrows}_f} \cup \bigcup_{a \in \mathcal{A}} [\bar{S}_a]^{\texttt{FKS}}_{\overset{\phi}{\leftrightarrows}_f}$.

  In order to show that $S_1 = S_2$, it suffices to establish that there are isomorphisms between $[S]^{\texttt{FTS}}_{\overset{\phi}{\leftrightarrows}_f}$ and $[S]^{\texttt{FKS}}_{\overset{\phi}{\leftrightarrows}_f}$, and between $[\bar{S}_a]^{\texttt{FKS}}_{\overset{\phi}{\leftrightarrows}_f}$ and $\overline{[S]^{\texttt{FTS}}_{\overset{\phi}{\leftrightarrows}_f}}|_a$, for all $a \in \mathcal{A}$.

  Using Theorem 7.11 we immediately obtain $[S]^{\texttt{FTS}}_{\overset{\phi}{\leftrightarrows}_f} = [S]^{\texttt{FKS}}_{\overset{\phi}{\leftrightarrows}_f}$.

  Furthermore, we can combine Theorem 7.11 with Lemma 7.10 to derive that $\texttt{fks}(fts) \models \bar{s}_a \overset{\phi}{\leftrightarrows}_f \bar{t}_a$ iff $fks \models s \overset{\phi}{\leftrightarrows}_f t$, for all $s, t \in S$ such that $\bar{s}_a, \bar{t}_a \in \bar{S}$, for all $a \in \mathcal{A}$, from which it follows that $[\bar{S}_a]^{\texttt{FKS}}_{\overset{\phi}{\leftrightarrows}_f} = \overline{[S]^{\texttt{FTS}}_{\overset{\phi}{\leftrightarrows}_f}}|_a$

- Since feature bisimulation reduction does not modify the set of atomic propositions, we immediately obtain $AP_1 = \mathcal{A} \cup \{\bot\} = AP_2$. We will refer to AP_1 and AP_2 as AP.

- The transition constraint function of $\texttt{fks}(fts)$ is $\theta' : S' \times S' \to \mathbb{B}(\mathcal{F})$, which is defined such that, for all $s, s' \in S'$, $t \in S$ and $a \in \mathcal{A}$:

$$\theta'(s,s') = \begin{cases} \mathsf{true} & \text{if } s = \bar{t}_a \text{ and } s' = t \\ \theta(s,a,t) & \text{if } s \in S \text{ and } s' = \bar{t}_a \\ \theta(s,\tau,s') & \text{if } s, s' \in S \\ \mathsf{false} & \text{otherwise} \end{cases}$$

We have $\theta_1 : S_1 \times S_1 \to \mathbb{B}(\mathcal{F})$ is such that, for all $C_1, C_2 \in S_1$:

$$\theta_1(C_1,C_2) = \bigvee\{\, \theta'(s,t) \mid s \in C_1 \wedge t \in C_2 \,\}.$$

The transition constraint function of $\overset{\phi}{\underset{f}{\leftrightarrow}}$-$\min_{\texttt{FTS}}(fts)$ is $\theta'' : [S]^{\texttt{FTS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}} \times \mathcal{A}_\tau \times [S]^{\texttt{FTS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}} \to \mathbb{B}(\mathcal{F})$, which is defined such that, for all $C_1, C_2 \in [S]^{\texttt{FTS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}}$, and all $\alpha \in \mathcal{A}_\tau$:

$$\theta''(C_1,\alpha,C_2) = \bigvee\{\, \theta(s,\alpha,t) \mid s \in C_1 \wedge t \in C_2 \,\}.$$

We have $\theta_2 : S_2 \times S_2 \to \mathbb{B}(\mathcal{F})$ is such that, for all $C_1, C_2 \in S_2$, $C_3 \in [S]^{\texttt{FTS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}}$, and for all $a \in \mathcal{A}$.

$$\theta_2(C_1,C_2) = \begin{cases} \mathsf{true} & \text{if } C_1 = \bar{C}_{3a} \text{ and } C_2 = C_3 \\ \theta''(C_1,a,C_3) & \text{if } C_1 \in [S]^{\texttt{FTS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}} \text{ and } C_2 = \bar{C}_{3a} \\ \theta''(C_1,\tau,C_2) & \text{if } C_1, C_2 \in [S]^{\texttt{FTS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}} \\ \mathsf{false} & \text{otherwise} \end{cases}$$

We will now show that $\theta_1 = \theta_2$. Pick two states $C_1, C_2 \in S_1$. We distinguish four cases:

1. $C_1, C_2 \in [S]^{\texttt{FKS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}}$. In this case the images of C_1 and C_2 in S_2 are C_1 and C_2, respectively. Since $C_1, C_2 \in [S]^{\texttt{FKS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}}$ we have $s, t \in S$, for all $s \in C_1$ and $t \in C_2$. We find
$$\theta_1(C_1,C_2) = \bigvee\{\, \theta(s,\tau,t) \mid s \in C_1 \wedge t \in C_2 \,\} = \theta_2(C_1,C_2).$$
2. $C_1 \in [S]^{\texttt{FKS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}}$ and $C_2 \in [\bar{S}_a]^{\texttt{FKS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}}$, for some $a \in \mathcal{A}$. Let $C_2' \subseteq S$ denote the set of states in S corresponding to C_2. That is, $C_2' = \{t \in S \colon \bar{t}_a \in C_2\}$. Now let $C_3 \in [S]^{\texttt{FKS}}_{\overset{\phi}{\underset{f}{\leftrightarrow}}}$ be such that $C_2' \subseteq C_3$. Note that such C_3 exists by Lemma 7.10.
In this case the images of C_1 and C_2 in S_2 are C_1 and $\bar{C}_{3a}$, respectively. We find
$$\theta_1(C_1,C_2) = \bigvee\{\, \theta(s,a,t) \mid s \in C_1 \wedge \bar{t}_a \in C_2 \,\}$$

and

$$\theta_2(C_1, \bar{C}_{3a}) = \theta''(C_1, a, C_3) = \bigvee\{\, \theta(s, a, t) | s \in C_1 \wedge t \in C_3 \,\}.$$

From which we conclude that $\theta_1(C_1, C_2) = \theta_2(C_1, \bar{C}_{3a})$.

3. $C_1 \in [\bar{S}_a]^{\mathtt{FKS}}_{\underset{\phi}{\leftrightarrow}_f}$, for some $a \in \mathcal{A}$ and $C_2 \in [S]^{\mathtt{FKS}}_{\underset{\phi}{\leftrightarrow}_f}$. Let $C_1' \subseteq S$ denote the set of states in S corresponding to C_1. That is, $C_1' = \{t \in S \colon \bar{t}_a \in C_1\}$. Now let $C_3 \in [S]^{\mathtt{FKS}}_{\underset{\phi}{\leftrightarrow}_f}$ be such that $C_1' \subseteq C_3$. Note that such C_3 exists by Lemma 7.10.
   In this case the images of C_1 and C_2 in S_2 are $\bar{C}_{3a}$ and C_2, respectively.
   We distinguish two cases:
   (a) $C_3 = C_2$. We find
   $$\theta_1(C_1, C_2) = \mathsf{true} = \theta_2(\bar{C}_{3a}, C_2).$$
   (b) $C_3 \neq C_2$. We find
   $$\theta_1(C_1, C_2) = \mathsf{false} = \theta_2(\bar{C}_{3a}, C_2).$$
   In both cases we find that that $\theta_1(C_1, C_2) = \theta_2(\bar{C}_{3a}, C_2)$.
4. $C_1 \in [\bar{S}_a]^{\mathtt{FKS}}_{\underset{\phi}{\leftrightarrow}_f}$ and $C_2 \in [\bar{S}_b]^{\mathtt{FKS}}_{\underset{\phi}{\leftrightarrow}_f}$, for some $a, b \in \mathcal{A}$. Let $C_1' \subseteq S$ denote the set of states in S corresponding to C_1. That is, $C_1' = \{t \in S \colon \bar{t}_a \in C_1\}$. Now let $C_3 \in [S]^{\mathtt{FKS}}_{\underset{\phi}{\leftrightarrow}_f}$ be such that $C_1' \subseteq C_3$. Note that such C_3 exists by Lemma 7.10. Furthermore, let $C_2' \subseteq S$ denote the set of states in S corresponding to C_2. That is, $C_2' = \{t \in S \colon \bar{t}_a \in C_2\}$. Now let $C_4 \in [S]^{\mathtt{FKS}}_{\underset{\phi}{\leftrightarrow}_f}$ be such that $C_2' \subseteq C_4$. Note that such C_4 exists by Lemma 7.10.
   In this case the images of C_1 and C_2 in S_2 are $\bar{C}_{3a}$ and $\bar{C}_{4b}$, respectively. We find
   $$\theta_1(C_1, C_2) = \mathsf{false} = \theta_2(\bar{C}_{3a}, \bar{C}_{4b}).$$

We conclude that $\theta_1 = \theta_2$.

- For each state $C \in [S]^{\mathtt{FKS}}_{\underset{\phi}{\leftrightarrow}_f}$ we have $L_1(C) = \{\bot\}$, by definition of `fks`. The image of C in S_2 is C, and we also have $L_2(C) = \{\bot\}$. For each state $C \in [\bar{S}_a]^{\mathtt{FKS}}_{\underset{\phi}{\leftrightarrow}_f}$, for some $a \in \mathcal{A}$, we have $L_1 = \{a\}$, by definition of `fks`. Let $C' \subseteq S$ denote the set of states in S corresponding to C. That is, $C' = \{t \in S \colon \bar{t}_a \in C\}$. Now let $C'' \in [S]^{\mathtt{FKS}}_{\underset{\phi}{\leftrightarrow}_f}$ be such that $C' \subseteq C''$. Note that such C'' exists by Lemma 7.10.
  The image of C in S_2 is now $\bar{C''}_a$, and we find $L_2(\bar{C''}_a) = \{a\}$. Hence we conclude that $L_1 = L_2$.
- The embedding `fks` does not change the initial state, and hence we have $s_{*1} = [s_*]_{\simeq_{S_1}}$, and $s_{*2} = [s_*]_{\simeq_{S_2}}$. Since $S_1 = S_2$, it follows that $s_{*1} = s_{*2}$.

We conclude that indeed $\overset{\phi}{\leftrightarrow}_f\text{-min}_{\mathtt{FKS}} \circ \mathtt{fks} = \mathtt{fks} \circ \overset{\phi}{\leftrightarrow}_f\text{-min}_{\mathtt{FTS}}$. □

Using the lemma, we straightforwardly obtain the desired result:

Theorem 7.13. *We have* $\overset{\phi}{\leftrightarrows}_f\text{-min}_{\texttt{FTS}} = \texttt{fks}^{-1} \circ \overset{\phi}{\leftrightarrows}_f\text{-min}_{\texttt{FKS}} \circ \texttt{fks}$*, for all* $\phi \in \mathbb{B}(\mathcal{F})$*.*

Proof. Lemma 7.11 gives us

$$\texttt{fks}^{-1} \circ \overset{\phi}{\leftrightarrows}_f\text{-min}_{\texttt{FKS}} \circ \texttt{fks} = \texttt{fks}^{-1} \circ \texttt{fks} \circ \overset{\phi}{\leftrightarrows}_f\text{-min}_{\texttt{FTS}}.$$

Using Theorem 7.12 we obtain our desired conclusion:

$$\texttt{fks}^{-1} \circ \overset{\phi}{\leftrightarrows}_f\text{-min}_{\texttt{FKS}} \circ \texttt{fks} = \overset{\phi}{\leftrightarrows}_f\text{-min}_{\texttt{FTS}}.$$

□

Unfortunately, minimality modulo coherent feature bisimilarity is not preserved by the embedding `fks`. An example of this is shown in Figure 7.10.

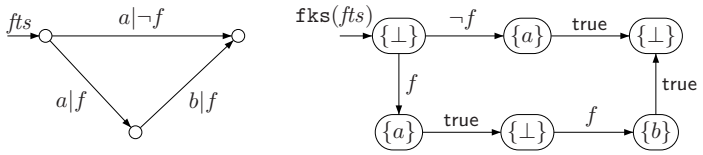


Figure 7.10: An FTS *fts* that is minimal modulo coherent feature bisimulation, and its corresponding FKS `fks`(*fts*), of which the two states with label $\{a\}$ are coherent feature bisimilar.

We see that the two states with label $\{a\}$ in the FKS are coherent feature bisimilar. The issue is that these states are reachable for less products than their target states. Hence, it is now possible to identify two states representing transitions from the FTS, without identifying their target states. This is not possible in the single-product setting, and hence the embedding `ks` does preserve minimality modulo bisimulation.

In an attempt to solve this, we will define a slightly adapted coherent quotient, specifically developed to be applied to reversible FKS.

7.2.3 The coherent$^+$-quotients for FKS

We observed that minimality modulo coherent bisimulation for FTS is not being preserved by the embedding `fks`. Also, the coherent quotient for FKS does not preserve reversibility of FKS. Hence, we propose an alternative for this quotient.

In a reversible FKS we can divide the state space in two subsets: the states with label $\{\bot\}$, which we will refer to as $S_\bot$, and the states with a label that does not contain $\{\bot\}$, which we will refer to as $S_\mathcal{A}$. Each state from $s \in S_\mathcal{A}$ has exactly one outgoing transition to a state from $s' \in S_\bot$, which is labeled with the feature expression `true`.

We refer to s' as the *target* of s, from now on denoted as $t(s) = s'$. Using this target function we redefine the reachability function ϱ.

Definition 7.12. *Given a reversible FKS $fks = (S, AP, \theta, L, s_*)$ with reachability function ϱ. The reachability$^+$ function ϱ^+ of fks is defined such that, for all $s \in S$:*

$$\varrho^+(s) = \begin{cases} \varrho(t(s)) & \text{if } \bot \notin L(s) \\ \varrho(s) & \text{if } \bot \in L(s) \end{cases}$$

Note that $\varrho(s) \Rightarrow_{\mathcal{P}} \varrho^+(s)$, for all $s \in S$. This is the case since each $s \in S_{\mathcal{A}}$ we have $s \xrightarrow{\mathsf{true}} t(s)$ and hence $\varrho(s) \Rightarrow_{\mathcal{P}} \varrho(t(s))$. Using the new reachability function we give an adapted definition of coherent bisimilarity.

Definition 7.13. *Let $fks = (S, AP, \theta, L, s_*)$ be a reversible FKS with reachability$^+$ function ϱ^+. We say two states $s, t \in S$ are coherent$^+$ feature bisimilar, denoted by $fks \models s \leftrightarrows_{cf+} t$ if and only if $fks \models s \xleftrightarrow{\varrho^+(s) \wedge \varrho^+(t)}_f t$.*

We use coherent$^+$ feature bisimilarity to define a set of quotients for reversible FKS.

Definition 7.14. *Let $fks = (S, AP, \theta, L, s_*)$ be a reversible FKS with reachability$^+$ function ϱ^+. A coherent$^+$ bisimulation quotient of fks is an FKS $fks' = (S', AP, \theta', L', s'_*)$, such that*

- *$S' \subseteq \bigcup_{C \in \langle S \rangle \underline{\leftrightarrows}_{cf+}} 2^C$, such that*
  1. *$\bigcup S' = S$;*
  2. *$C_1 \cap C_2 = \emptyset$, for all distinct $C_1, C_2 \in S'$;*
  3. *$|S'|$ is minimal.*
- *for all $C \in S'$, $L'(C) = L(s)$ for all $s \in C$.*
- *$\theta' \colon S' \times S'$ is constructed such that, for all $C_1, C_2 \in S'$:*

$$\theta'(C_1, C_2) = \begin{cases} \mathsf{true} & \text{if } \bot \notin L'(C_1) \text{ and } \exists (s,t) \in C_1 \times C_2 \colon s \to t \\ \bigvee \{\, \theta(s,t) \wedge \varrho^+(s) \mid (s,t) \in C_1 \times C_2 \,\} & \text{otherwise} \end{cases}$$

- *$s'_* = [s_*]_{\simeq_{S'}}$.*

The function that yields the set of all coherent$^+$ feature bisimulation quotients of a reversible FKS is denoted by $\underline{\leftrightarrows}_{cf+}\text{-min}_{\mathtt{FKS}}$.

Note that the value of the transition constraint function is defined to be true for all outgoing transitions of classes not labeled with $\bot$. This is done to preserve the third requirement of reversibility for FKS (Definition 7.10), stating that each state s not labeled with $\bot$ must have exactly one target state $t(s)$ such that $s \xrightarrow{\mathsf{true}} t(s)$.

From Lemma 7.10 it follows that two states $s, t \in S_{\mathcal{A}}$ of a reversible FKS *fks* with state space S, such that $L(s) = L(t)$, are coherent$^+$ feature bisimilar if and only if their respective target states $t(s)$ and $t(t)$ are coherent$^+$ feature bisimilar. It follows that the set of target states $\{t(s) \mid s \in C\}$ of each class $C \in \langle S_{\mathcal{A}} \rangle \underline{\leftrightarrows}_{cf+}$ is a subset of some class $C' \in \langle S_{\bot} \rangle \underline{\leftrightarrows}_{cf+}$. Using this knowledge it seems like a coherent$^+$ feature bisimulation quotient *fks'* of *fks* indeed satisfies the third requirement of reversibility for FKS. However, here we are not taking into account that $\langle S \rangle \underline{\leftrightarrows}_{cf+}$ differs from the actual state space of *fks'*. Consider the example shown in Figure 7.11.

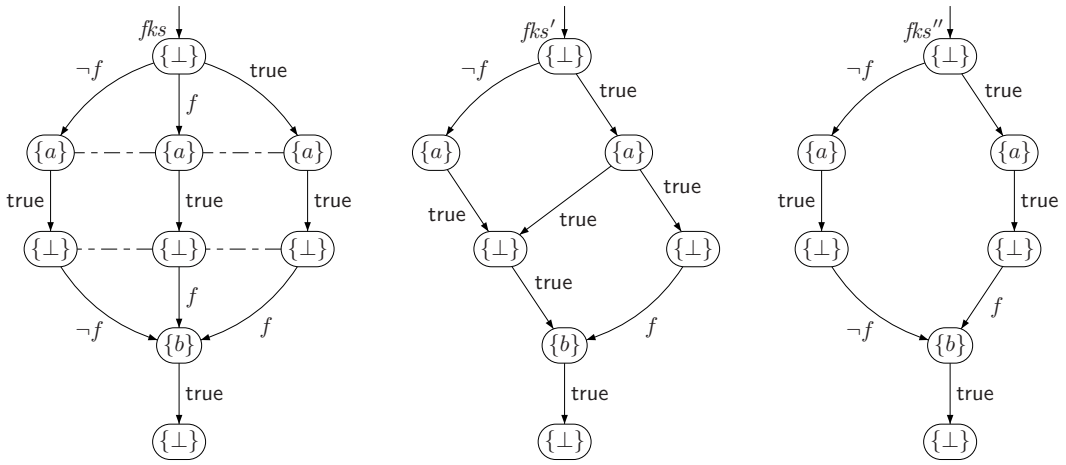


Figure 7.11: A reversible FKS *fks* and two of its coherent$^+$ feature bisimulation quotients *fks*′ and *fks*″.

The FKS *fks* in Figure 7.11 has multiple coherent$^+$ feature bisimulation quotients. Some of these quotients, for example *fks*′, do not respect reversibility for FKS, while others such as *fks*″ do. From this example we conclude that not all coherent$^+$ feature bisimulation quotients of a reversible FKS are reversible, but it seems that there is always at least one reversible coherent$^+$ feauture bisimulation quotient. However, we found that this is also not true. Consider the example shown in Figure 7.12.

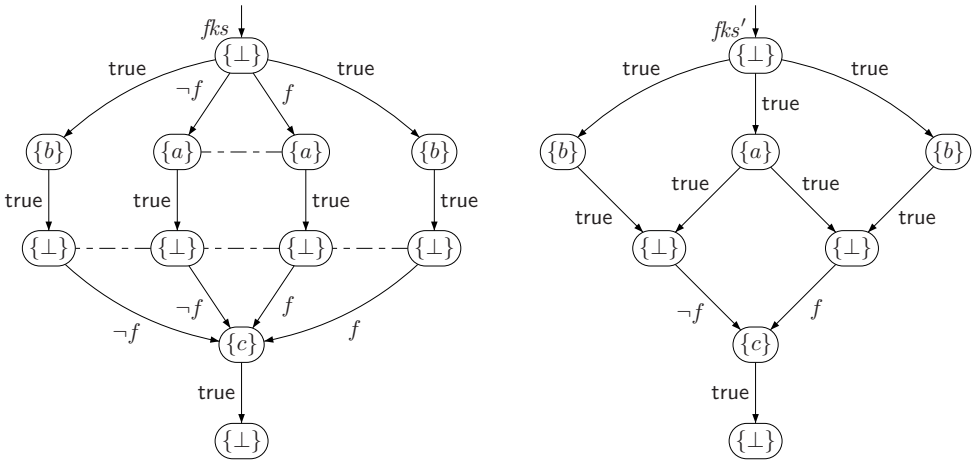


Figure 7.12: A reversible FKS *fks* and its only coherent$^+$ feature bisimulation quotient *fks*′.

The reversible FKS *fks* in Figure 7.12 has only one coherent$^+$ feature bisimulation quotient, which is not reversible. Even worse, it is not feature bisimilar to *fks*. The problem is that the classes formed in $S_{\mathcal{A}}$ correspond to *subsets* of the classes formed in $S_{\perp}$. Therefore it may happen, as shown in Figure 7.12, that the coherent$^+$ feature bisimulation quotients of an FKS consist of non-corresponding classes on these two levels, which violates the third condition of

reversibility for FKS. A possible solution could be to weaken this condition. However, if we do this we are not able to prove anymore that the naive feature bisimulation quotient for FKS preserves reversibility. Therefore we are not sure how to resolve this problem, and we leave it to future work to determine whether it is possible to calculate a coherent feature bisimulation quotient of an FTS by minimizing its corresponding FKS.

We again conclude by summarizing the obtained results. As for the embedding to an event-based model, we started in Section 7.2.1 by proposing a straightforward generalization of the embedding `ks` to the product-family level, resulting in the embedding `fks`. We proved that this embedding preserves feature bisimulation in Theorem 7.11. We defined the embedding $\texttt{fks}^{-1}$, and showed that it is the left inverse of `fks` (Theorem 7.12).

In Section 7.2.2 we proved with Theorem 7.13 that we can perform naive feature bisimulation minimization on an FTS by performing naive feature bisimulation minimization on its corresponding FKS.

Unfortunately, as with the embedding `fts` we found that `fks` does not preserve minimality modulo coherent feature bisimulation. Furthermore we found that the coherent quotient for FKS does not preserve reversibility of FKS. We made an attempt to solve the problem by proposing a coherent$^+$ feature bisimulation quotient for reversible FKS in Section 7.2.3, but also this quotient does not preserve reversibility. Since we are not sure how to resolve this problem, we leave it to future work to determine whether it is possible to calculate a coherent feature bisimulation quotient of an FTS by minimizing its corresponding FKS.

Chapter 8

Experimental evaluation

In order to evaluate the practical usefulness of the developed algorithm to perform complete coherent branching feature bisimulation reduction, we performed a small case study. As a toy example we extended the example SPL of a coffee vending machine described in [2, 3, 5, 6] with a soup component running in parallel. The complete SPL consists of 18 features and 118 products and the FTS modeling it contains 182 states and 691 transitions.

The details of this SPL are described in Section 8.1. Section 8.2 describes the developed SPL toolset, Section 8.3 describes the performed experiments, and Section 8.4 presents the results.

8.1 The coffee-soup machine SPL

The coffee-soup machine SPL is an extension of the coffee vending machine described in [2, 3, 5, 6] with a soup component running in parallel with the usual beverage component. It has the following list of functional requirements:

- Each product contains a beverage component. Optionally, also a soup component is present.
- Initially, either a euro must be inserted, exclusively for European products, or a dollar must be inserted, exclusively for Canadian products. The money can be inserted in either of the components.
- Optionally, money inserted in a component can be retrieved via a cancel button, after which money can be inserted in this component anew.
- If money was inserted in the beverage component, the user has to choose whether (s)he wants sugar, by pressing one of two buttons, after which (s)he can select a beverage.
- The choice of beverage (coffee, tea, cappuccino) varies, but coffee must be offered by all products whereas cappuccino may be offered solely by European products.
- Optionally, a ringtone may be rung after delivering a beverage. However, a ringtone must be rung by all products offering cappuccino.
- After the beverage is taken, money can be inserted again in the beverage component.

- If money was inserted in the soup component, the user has to choose a type of soup (chicken, tomato, pea). The types of soup offered vary, but at least one type must be offered by all products with a soup component.

- The soup component does not contain cups to serve the soup in. Hence, the user has to place a cup to pour the soup in. Optionally, a cup detector may be present in the soup component. It is required that all Canadian products with a soup component are equipped with a cup detector.

- If cup detection is present, the chosen type of soup will only be delivered after a cup has been detected by the soup component. However, the cup detector may fail to detect an already placed cup, after which the user will have to place it again. If a cancel option is available, the user may cancel the order as long as no cup has been detected.

- If cup detection is not present, the soup will be delivered immediately after a type of soup was chosen, regardless of whether a cup was placed. If no cup was placed there will be no soup to take.

- Optionally, a ringtone (shared with the beverage component) may be rung after delivering soup.

- If a cup was present, money can be inserted again in the soup component after the soup is taken. If no cup was present, money can be inserted again immediately after the soup has been delivered.

These yield the attributed feature model in Figure 8.1 and the behavioral models in Figures 8.2 and 8.3.

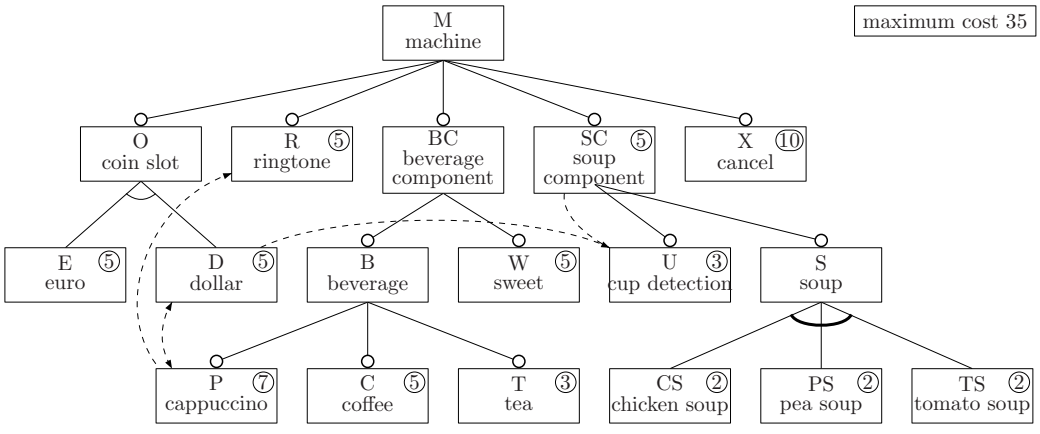


Figure 8.1: Feature model of family of coffee vending machines

In the attributed feature model, mandatory (core) features are marked by a closed bullet, optional features by an open one. Exactly one of the features E and D is selected, while at least one of the features CS, PS and TS is selected. As to cross-tree constraints, features P and D exclude each other, feature P requires feature R, and the simultaneous selection of features D and SC requires feature U. The value of the cost attribute of the concrete features is put inside a small circle (i.e. $cost(X) = 10$). Finally, as an additional constraint, we require that the total costs of all selected features does not exceed the threshold 35.

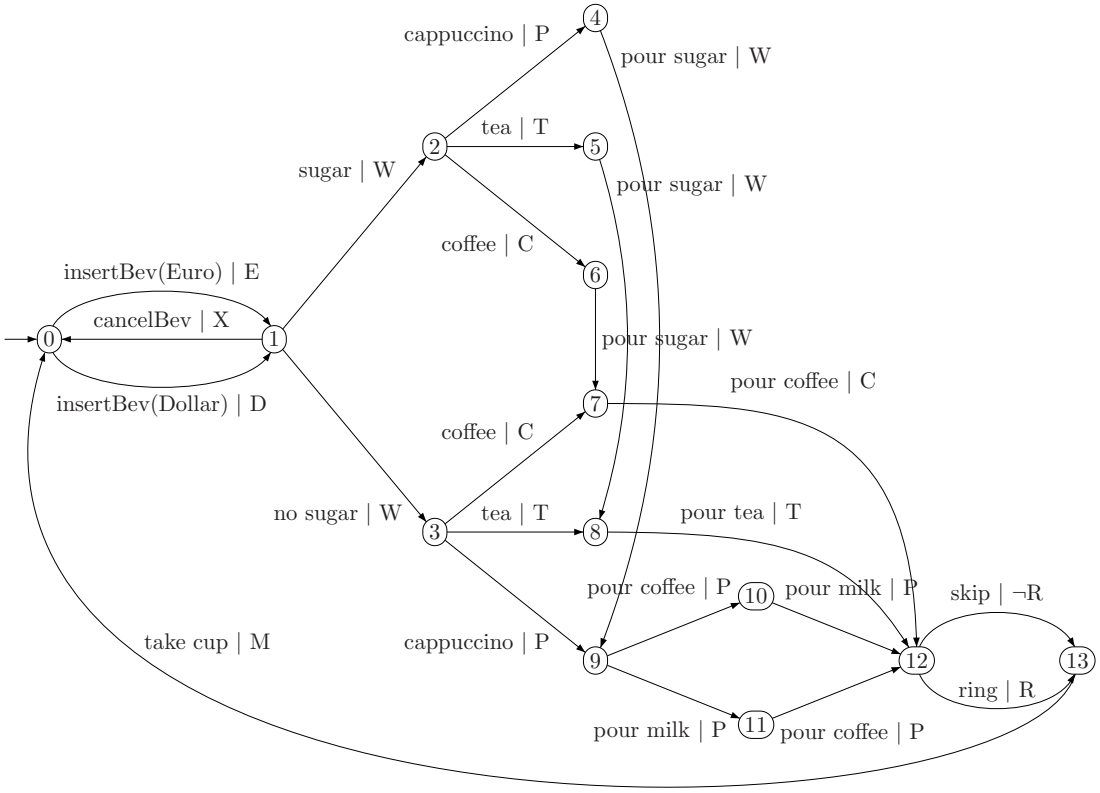


Figure 8.2: FTS of beverage component

The FTS of the beverage component contains 14 states and 23 transitions and that of the soup component contains 13 states and 28 transitions, for a total of 182 states and 691 transitions in parallel composition.

8.2 The SPL toolset

A toolset was developed to define, abstract and reduce SPL. The SPL are defined using FD and FTS, and a conversion tool was created to convert SPL specifications to mCRL2 specifications, which allows for automated model checking using the mCRL2 toolset[1].

The developed SPL toolset consists of the following tools:

[1] `http://www.mcrl2.org`

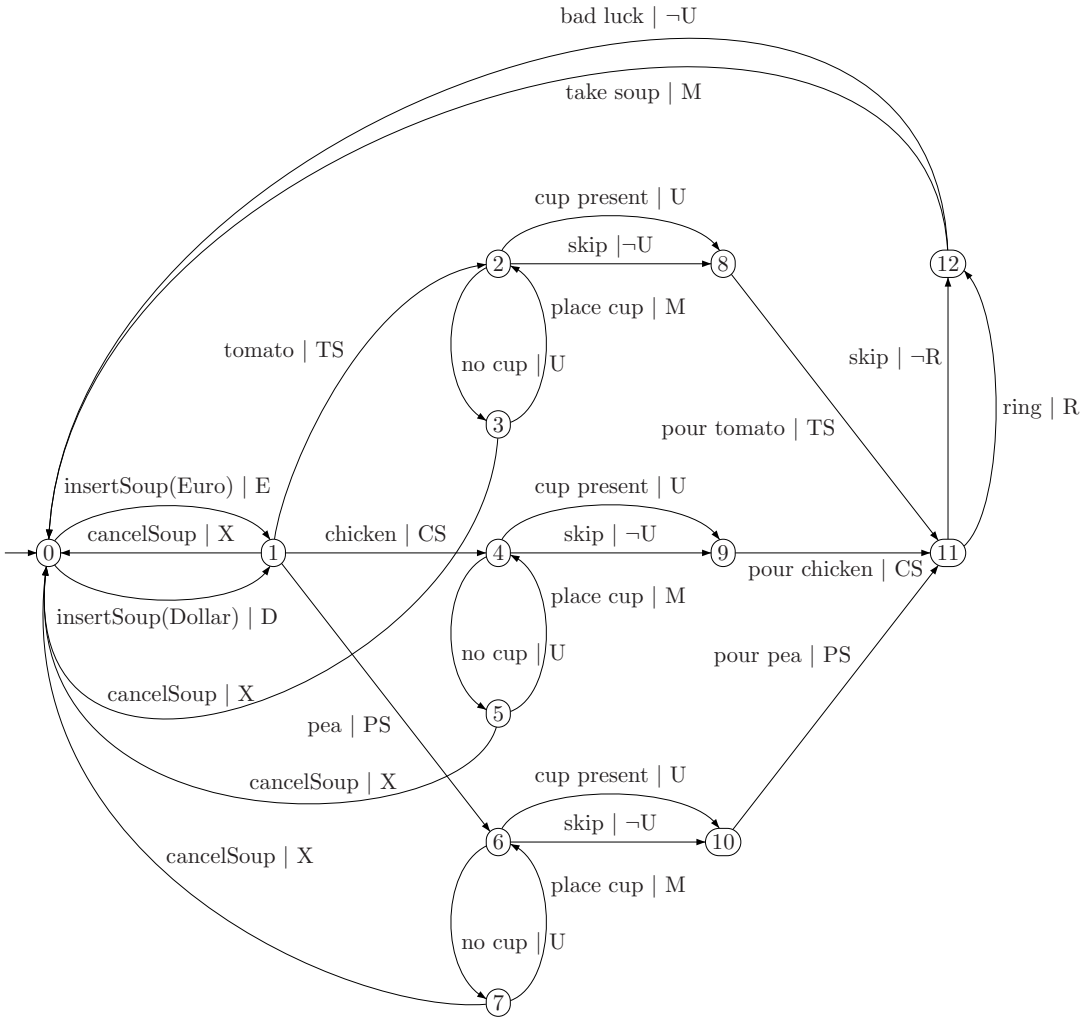


Figure 8.3: FTS of soup component

- `parseFD:` Calculates products and feature expressions for a given FD.
- `abstractSPL:` Abstracts from certain parts of the FTS specifying the behavior of a given SPL.
- `BFBreduction:` Reduce an the FTS specifying the behavior of a given SPL modulo complete coherent branching feature bisimulation.
- `spl2mcrl2:` Converts an SPL specification to an mCRL2 specification.
- `fts2aut:` Converts an FTS to an LTS in Aldebaran format[2].
- `projectSPL:` Reduces the FFTS specifying the behavior of a given SPL to a given product.
- `splprod2mcrl2:` Converts an SPL-product specification to an mCRL2 specification.

[2] `http://www.mcrl2.org/release/user_manual/language_reference/lts.html#aldebaran-format`

- `lts2aut`: Converts an LTS to an LTS in Aldebaran format.

Appendix F contains a detailed user manual of the FTS toolset.

8.3 Experiments

We used the SPL and mCRL2 toolsets to verify 12 properties against the coffee-soup machine SPL. These properties are listed next, together with their formalization in the mCRL2 variant of the modal μ-calculus.

1. If a coffee is ordered and the system does not deadlock, then eventually coffee is poured:
   $[\, true* \,.\, coffee \,]\, (\text{mu}\, X.\, [\, !\, pour\_coffee \,]\, X)$
2. The SPL is deadlock-free:
   $[\, true* \,]\, \langle true \rangle\, true$

3a. A machine that accepts Euros does not accept Dollars:
   $[true* .(insertBev(Euro) \mid\mid insertSoup(Euro)).true* .$
   $(insertBev(Dollar) \mid\mid insertSoup(Dollar))]\, false$

3b. A machine that accepts Dollars does not accept Euros:
   $[true* .(insertBev(Dollar) \mid\mid insertSoup(Dollar)).true* .$
   $(insertBev(Euro) \mid\mid insertSoup(Euro))]\, false$

4a. A cup can only be taken out of the beverage component after a beverage was ordered:
   $[\, (!\, coffee \;\&\&\; !\, tea \;\&\&\; !\, cappuccino)* \,.\, take\_cup \,]\; false$

4b. A cup can only be taken out of the soup component after soup was ordered:
   $[\, (!\, tomato \;\&\&\; !\, chicken \;\&\&\; !\, pea)* \,.\, take\_soup]\; false$

5a. If a beverage is ordered and the system does not deadlock, then eventually the beverage is canceled or a cup is taken out of the beverage component:
   $[\, true* \,.\, (\, coffee \mid\mid tea \mid\mid cappuccino) \,]\, (\text{mu}\, X.\, [\, (!\, cancelBev \;\&\&\; !\, take\_cup) \,]\, X)$

5b. If soup is ordered and the system does not deadlock, then eventually the soup is canceled, a cup is taken out of the soup component or the customer has bad luck:
   $[\, true*.\, (tomato \mid\mid chicken \mid\mid pea) \,]\, (\text{mu}\, X.\, [\, (!\, cancelSoup \;\&\&\; !\, take\_soup \;\&\&\; !\, bad\_luck) \,]\, X)$

6. If the machine has a soup component, then a beverage can be ordered without inserting more money after soup was ordered:
   $[\, true* \,.\, (insertSoup(Euro) \mid\mid insertSoup(Dollar)) \,]\, \langle true* \,.\, (tomato \mid\mid chicken \mid\mid pea).$
   $(!\, insertBev(Euro) \;\&\&\; !\, insertBev(Dollar))* \,.\, (coffee \mid\mid tea \mid\mid cappuccino) \rangle\, true$

7a. A beverage cannot be ordered without inserting more money if a previous beverage order is still pending:
   $[true* .(coffee \mid\mid tea \mid\mid cappuccino).(!insertBev(Dollar) \;\&\&\; !insertBev(Euro))* .(coffee \mid\mid$
   $tea \mid\mid cappuccino)]\; false$

7b. Soup cannot be ordered without inserting more money if a soup order is pending:
   $[\, true*.\, (tomato \mid\mid chicken \mid\mid pea).\, (!\, insertSoup(Dollar) \;\&\&\; !\, insertSoup(Euro))*.\, (tomato \mid\mid$
   $chicken \mid\mid pea) \,]\; false$

8. In a machine with cup detection, witnessed by the possibility to perform a *cup_present* action, soup can only be poured after detecting a cup:
$[\, true * .\, cup\_present\,][true * .\, (take\_soup \,||\, bad\_luck).\, (!\, cup\_present) * .$
$(pour\_tomato \,||\, pour\_chicken \,||\, pour\_pea)]\; false$

We verified the properties both product-by-product and by using the FTS-based family approach described in [5, 6], and both with and without branching (feature) bisimulation minimization. For the approach with bisimulation we applied branching feature bisimulation to the FTS, resulting in a reduced FTS, which we projected to obtain the reduced LTS for each product. For the product-by-product approaches, generating the projections for all products is included in the computation time, and so is the time for bisimulation reduction in case of the approaches with bisimulation. To even out effects caused by other processes running whilst performing the experiments, all computation times are averaged over 5 runs.

8.4 Results

The verification results are shown in Table 8.1.

	PRODUCT-BY-PRODUCT				FTS-BASED FAMILY APPROACH			
	WITHOUT BISIMULATION		WITH BISIMULATION		WITHOUT BISIMULATION		WITH BISIMULATION	
	TIME (s)	RESULT	TIME (s)	RESULT	TIME (s)	RESULT	TIME (s)	RESULT
1	32.67	FALSE	29.50	TRUE	29.43	FALSE	1.45	TRUE
2	31.90	TRUE	32.42	TRUE	29.43	TRUE	25.77	TRUE
3a	32.83	TRUE	27.79	TRUE	41.14	TRUE	1.62	TRUE
3b	32.33	TRUE	27.80	TRUE	33.29	TRUE	1.55	TRUE
4a	31.47	TRUE	28.46	TRUE	11.13	TRUE	1.22	TRUE
4b	30.95	TRUE	28.59	TRUE	7.12	TRUE	1.52	TRUE
5a	33.21	FALSE	28.63	TRUE	37.01	FALSE	1.62	TRUE
5b	33.45	FALSE	29.10	TRUE	41.06	FALSE	2.10	TRUE
6	34.80	TRUE	29.65	TRUE	67.27	TRUE	8.32	TRUE
7a	33.19	TRUE	29.18	TRUE	40.78	TRUE	1.71	TRUE
7b	33.16	TRUE	28.32	TRUE	48.30	TRUE	1.70	TRUE
8	33.58	TRUE	29.04	TRUE	50.90	TRUE	2.95	TRUE
TOT	393.54		348.48		436.86		51.53	

Table 8.1: Experimental evaluation results (time in seconds)

Regarding the product-by-product approach, performing bisimulation reduction for the product LTS reduces the computation time by about 11%. For property 2 (*The SPL is deadlock-free*), the computation time with bisimulation is significantly larger than for other properties. In this case no actions can be abstracted from, and hence applying branching bisimulation does not significantly reduce the LTS.

We observe that the properties 1 (*If a coffee is ordered, it is eventually poured*), 5a (*If a beverage is ordered, then eventually it is canceled or a cup is taken*) and 5b (*If soup is ordered, then eventually it is canceled, a cup is taken or the customer has bad luck*) are false, but deemed true after applying bisimulation reduction. They state that some event eventually happens, which is not true in reality since the two components are running in parallel, and hence infinite loops exist that allow postponing that event indefinitely. Applying bisimulation reduction causes these loops to be abstracted from completely, making the properties true for the reduced system.

Now consider the FTS-based family approach. Without applying bisimulation reduction, the total computation time increases by about 11% with respect to the product-by-product approach. Hence, for this SPL, FTS-based verification with mCRL2 is not beneficial compared to regular enumerative verification. However, if we apply bisimulation reduction, then the FTS-based computation times decrease by $>85\%$. Note that in case fewer actions are involved in a property, it is possible to abstract from larger parts of the FTS, implying faster verification. This effect was much less in the product-by-product approach. Hence, the more local a property, the more beneficial it is to perform FTS-based family verification in combination with branching feature bisimulation reduction using mCRL2. Obviously, this observation needs to be confirmed by experimenting with different SPL, but based on this example the proposed techniques look rather promising.

Chapter 9

Conclusions

In this master thesis we have done research on the topic of model checking for software product lines. In times where people have an increasing interest in products that are fine tuned to fulfill their individual demands, the popularity of SPLE as a software engineering paradigm is growing. As SPLE is being applied for the development of safety critical systems, being able to perform model checking for the verification of SPL is desirable.

The starting point for our research was the work performed by Classen et al. in [10, 9]. They proposed a model based on transition systems to describe the behavior of SPL, which they called Featured Transition Systems. Although in theory this model contains both state information and event information, only the state information was used in the verification and abstraction techniques from [10, 9, 11]. Building on this work, Ter Beek & De Vink [5, 6] proposed a method to perform verification of Featured Transition Systems using the mCRL2 toolset [12]. However, here only the event information of the Featured Transition Systems was utilized, while the state information was ignored.
The research question and the research goal of this thesis are defined as follows:

1. *Are event-based models and state-based models on the product-family level equally expressive?*

2. *Develop an algorithm to perform state space reduction on product-family level models.*

To answer this question and fulfill this goal we first refined the model of Featured Transition Systems into a state-based variant as used by Classen et al., coined Feature Kripke Structure (FKS) for our purposes, and an event-based variant as used by Ter Beek & De Vink, which we call Feature Labeled Transition Systems (FTS). We immediately discovered a discrepancy between the two types of models, as the feature information was added to the transitions in both FKS and FTS, whereas one would expect this information to be added to the states in FKS. Even so, we continued by defining several equivalences for both types of models, based on strong bisimulation (both models), branching bisimulation (FTS), and divergence-blind stuttering equivalence (FKS). Here we noticed the effect of the discrepancy between the two models, as two states in an FTS are trivially considered equivalent for the empty set of product, whereas this is not the case for FKS.

Using the above equivalences, we proposed two types of quotients for the models. We refer to the first type as naive, as it naively requires equivalence for all possible products in order to perform minimization. This type of quotient corresponds to a simulation-equivalence based quotient for Featured Transition Systems as described in [11]. We refer to the second type of quotient as coherent. This type takes into account which products are capable of reaching which parts of the model, and hence results in smaller quotients.

As to partly fulfill the research goal of this thesis, we looked into the possibility of performing coherent branching feature bisimulation reduction on FTS. Unfortunately we found that performing coherent (branching) feature bisimulation reduction on FTS is NP-hard, as it can be reduced to the chromatic number problem. In spite of this negative result, we made an attempt to design an algorithm to perform coherent branching feature bisimulation reduction on FTS, based on the algorithm for branching bisimulation reduction proposed in [18]. However, we found that the proposed algorithm does not find an actual coherent branching feature bisimulation quotient. Instead, a refinement of coherent branching feature bisimulation is calculated, which we called complete coherent branching feature bisimulation.

To investigate the usefulness of the designed algorithm, we developed the SPL toolset. This toolset allows to define SPL using feature diagrams and FTS and to perform complete coherent branching feature bisimulation reduction on these FTS. Finally it allows to perform verification of the SPL using mCRL2, as proposed by Ter Beek & De Vink. We performed a small case study on a combined coffee and soup vending machine using the developed toolset. The results showed that a product-family based approach of verification in combination with complete coherent branching feature bisimulation reduction was much more effective than the regular enumerative method of verification, where each product is verified separately.

We continued by investigating whether event-based models and state-based models on the product-family level are equally expressive by defining and discussing embeddings between FTS and FKS. The embeddings we proposed are based on the embeddings by De Nicola & Vaandrager [13], between Labeled Transition Systems and Kripke Structures. Using the results from [26] on the properties of these embeddings we were able to show that our newly proposed embeddings preserve and reflect strong feature bisimulation. We also showed it is possible to perform naive feature bisimulation reduction in one world by using the embeddings and performing reduction in the other world. However, in order to make this work for coherent reduction, some small modifications had to be made to the embedding `fts` and the coherent quotient for FTS, since we found that a coherent strong feature bisimulation quotient in the event-based world does not completely correspond to coherent strong feature bisimulation in the action-based world. We did not succeed at all in performing coherent reduction in the event-based world by using embeddings to the action-based world. We believe this phenomenon is caused by the afore-mentioned discrepancy between the two models, but further research has to be performed in order to confirm this.

We have showed that we can perform naive state space reduction in one world by performing reduction in the other world. This observation indicates that event-based models and state-based models on the product-family level may be equally expressive, despite the observed difference between the two models. However, there is a clear discrepancy between the coherent quotients in both worlds. It remains to future work to investigate whether this is caused by the difference between the two types of models, or that it is a flaw of the coherent quotient itself.

We did however show that it is possible to perform coherent state space reduction in the action-based world by using the embeddings and performing a slightly adapted form of coherent reduction in the event-based world. This observation has practical use, as it implies that a variant of the designed algorithm to perform complete coherent strong feature bisimulation reduction will also be applicable to FKS. Furthermore, since the embeddings can be applied in polynomial time, we conclude that such state space reduction on FKS is NP-hard, just as it is for FTS.

9.1 Future work

In this thesis we have laid the theoretical foundations for embeddings between state-based and event-based systems for product families. A clear gap in the obtained results is the inability to perform coherent state-space reduction in the event-based world by performing reduction in the action-based world. Future work should determine if it is after all possible to perform this task, or if the discrepancy between the coherent bisimulation quotients in both worlds is a flaw of the coherent quotient itself.

As the obtained results in the thesis are theoretical, in order to evaluate the practical relevance of these embeddings an experimental evaluation should be performed in which reduction of product-family models is performed using these embeddings. Furthermore, in this thesis we only showed that the proposed embeddings preserve and reflect strong feature bisimulation. The work should be extended to investigate whether this also holds for branching feature bisimulation and divergence-blind stuttering feature equivalence, as this will tell us whether the proposed algorithm for complete coherent branching feature bisimulation reduction can also be applied to FKS.

In our case study, we noticed that several properties that were not satisfied by the original system were verified to be true after applying branching feature bisimulation reduction. This behavior occurs since branching bisimulation hides loops consisting of silent transitions. On the single-product level this problem is dealt with by refining the notion of branching bisimulation to divergence-sensitive branching bisimulation, which corresponds to stuttering equivalence in a state-based setting. In order to also apply these notions on the product-family level their definitions have to be lifted, as was done with the definitions of both strong and branching bisimulation. Subsequently, quotients on the product-family level have to be defined, as well as algorithms to calculate these quotients. Finally, to complete the picture it would have to be shown that the lifted equivalences are preserved and reflected by the embeddings we defined.

Even though the results of our work indicate that FKS and FTS may be equally expressive, we still found a discrepancy between the two types models which was visible at multiple points in our research. As suggested in Chapters 3 and 4, it may be possible to eliminate this discrepancy by defining an alternative state-based model, where the feature information is added to the states instead of to the transitions. It would be interesting to see how such an alternative model compares to FTS, and whether in this case the coherent feature bisimulation quotients of both worlds do correspond exactly.

Bibliography

[1] *Software Product Lines: Practices and Patterns*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[2] P. Asirelli, M.H. ter Beek, A. Fantechi, and S. Gnesi. Formal description of variability in product families. In *SPLC'11*, pages 130–139. IEEE, 2011.

[3] P. Asirelli, M.H. ter Beek, A. Fantechi, and S. Gnesi. A compositional framework to derive product line behavioural descriptions. In *ISoLA'12*, volume 7609 of *LNCS*, pages 146–161. Springer, 2012.

[4] T. Basten. Branching bisimilarity is an equivalence indeed! *Information Processing Letters*, 58:141–147, 1996.

[5] M.H. ter Beek and E.P. de Vink. Towards modular verification of software product lines with mCRL2. In *FMSPLE track at ISoLA'14*, volume 8802 of *LNCS*, pages 368–385. Springer, 2014.

[6] M.H. ter Beek and E.P. de Vink. Using mCRL2 for the analysis of software product lines. In *FormaliSE'14*, pages 31–37. IEEE, 2014.

[7] T. Belder, M. H. ter Beek, and E. P. de Vink. Coherent branching feature bisimulation. In *Proceedings 6th Workshop on Formal Methods and Analysis in SPL Engineering, FMSPLE 2015, London, UK, 11 April 2015.*, pages 14–30, 2015.

[8] M. C. Browne, E. M. Clarke, and O. Grumberg. Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59:115–131, 1988.

[9] A. Classen, M. Cordy, P. Heymans, P.-Y. Schobbens, A. Legay, and J.-F. Raskin. Featured transition systems: Foundations for verifying variability-intensive systems and their application to LTL model checking. *IEEE Transactions on Software Engineering*, 39:1069–1089, 2013.

[10] A. Classen, P. Heymans, P.-Y. Schobbens, A. Legay, and J.-F. Raskin. Model checking lots of systems: Efficient verification of temporal properties in software product lines. In *ICSE'10*, pages 335–344. ACM, 2010.

[11] M. Cordy, A. Classen, G. Perrouin, P.-Y. Schobbens, P. Heymans, and A. Legay. Simulation-based abstractions for software product-line model checking. In *ICSE'12*, pages 672–682. IEEE, 2012.

[12] S. Cranen, J.F. Groote, J.J.A. Keiren, F.P.M. Stappers, E.P. de Vink, W. Wesselink, and T.A.C. Willemse. An overview of the mCRL2 toolset and its recent advances. In *TACAS'13, LNCS 7795*, pages 199–213. Springer, 2013.

[13] R. De Nicola and F. W. Vaandrager. Action versus state based logics for transition systems. In *Semantics of Systems of Concurrent Processes, LITP Spring School on Theoretical Computer Science, La Roche Posay, France, April 23-27, 1990, Proceedings*, pages 407–419, 1990.

[14] R. De Nicola and F.W. Vaandrager. Three logics for branching bisimulation. *Journal of the ACM*, 42:458–487, 1995.

[15] R. J. van Glabbeek. The linear time – branching time spectrum II: The semantics of sequential systems with silent moves (extended abstract). In *CONCUR'93*, volume 715 of *LNCS*, pages 66–81. Springer, 1993.

[16] R. J. van Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43:555–600, 1996.

[17] R.J. van Glabbeek. The linear time – branching time spectrum (extended abstract). In *CONCUR'90*, volume 458 of *LNCS*, pages 278–297. Springer, 1990.

[18] J.F. Groote and F.W. Vaandrager. An efficient algorithm for branching bisimulation and stuttering equivalence. In *ICALP'90*, volume 443 of *LNCS*, pages 626–638. Springer, 1990.

[19] R. M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19:371–384, 1976.

[20] S. A. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[21] K. G. Larsen and B. Thomsen. A modal process logic. In *Proceedings of the Third Annual Symposium on Logic in Computer Science (LICS '88), Edinburgh, Scotland, UK, July 5-8, 1988*, pages 203–210, 1988.

[22] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[23] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[24] D. M. R. Park. Concurrency and automata on infinite sequences. In *Theoretical Computer Science, 5th GI-Conference, Karlsruhe, Germany, March 23-25, 1981*, pages 167–183, 1981.

[25] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering - Foundations, Principles, and Techniques*. Springer, 2005.

[26] M. A. Reniers, R. Schoren, and T. A. C. Willemse. Results on embeddings between state-based and event-based systems. *The Computer Journal*, 57:73–92, 2014.

[27] P.-Y. Schobbens, P. Heymans, and J.-C. Trigaux. Feature diagrams: A survey and a formal semantics. In *14th IEEE International Conference on Requirements Engineering (RE 2006), 11-15 September 2006, Minneapolis/St.Paul, Minnesota, USA*, pages 136–145, 2006.

Appendices

Appendix A

Full proofs for Chapter 3

This appendix contains the full proofs that were omitted in Chapter 3.

Lemma 3.1. *Let* $fks_i = (S_i, AP, \theta_i, L_i, s_{*\,i})$*, for* $i \in \{1, 2\}$*, be two FKS, and let* $P \in \mathcal{P}$ *be a product. We have:*

$$(fks_1 \uplus fks_2)|_P = fks_1|_P \uplus fks_2|_P$$

Proof. Let $P \in \mathcal{P}$ be a product and let $fks_i = (S_i, AP, \theta_i, L_i, s_{*\,i})$, for $i \in \{1, 2\}$, be two FKS. Let the KS $(fks_1 \uplus fks_2)|_P = (S, AP, \rightarrow, L, s_*)$ be the projection to P of their disjoint union, and let the KS $fks_1|_P \uplus fks_2|_P = (S', AP, \rightarrow', L', s'_*)$ be the disjoint union of their projections to P. We establish the isomorphism by proving that there are isomorphisms between S and S', $\rightarrow$ and $\rightarrow'$, L and L', and s_* and s'_*.

- The set of states is unchanged under projection, and hence $S = S_1 \uplus S_2 = S'$.
- Let the transition constrained function θ' of $fks_1 \uplus fks_2$ be as defined in Definition 3.6.

  We have

$$\begin{aligned} \rightarrow &= \{\ t \in S_1 \uplus S_2 \times S_1 \uplus S_2 \mid P \models \theta'(t)\ \} \\ &= \{\ t \in S_1 \times S_1 \mid P \models \theta_1(t)\ \} \uplus \{\ t \in S_2 \times S_2 \mid P \models \theta_2(t)\ \} \\ &= \rightarrow' \end{aligned}$$

- The state labeling function is unchanged under projection, and hence $L = L_1 \uplus L_2 = L'$.
- The initial state is unchanged under projection, and hence $s_* = s_{*\,1} = s'_*$.

□

Lemma 3.2. *Let* $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*\,i})$*, for* $i \in \{1, 2\}$*, be two FTS, and let* $P \in \mathcal{P}$ *be a product. We have:*

$$(fts_1 \uplus fts_2)|_P = fts_1|_P \uplus fts_2|_P$$

Proof. Let $P \in \mathcal{P}$ be a product and let $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*i})$, for $i \in \{1, 2\}$, be two FTS. Let the LTS $(fts_1 \uplus fts_2)|_P = (S, \mathcal{A}, \rightarrow, s_*)$ be the projection to P of their disjoint union, and let the LTS $fts_1|_P \uplus fts_2|_P = (S', \mathcal{A}, \rightarrow', s'_*)$ be the disjoint union of their projections to P. We establish the isomorphism by proving that there are isomorphisms between S and S', $\rightarrow$ and $\rightarrow'$, and s_* and s'_*.

- The set of states is unchanged under projection, and hence $S = S_1 \uplus S_2 = S'$.
- Let the transition constrained function θ' of $fts_1 \uplus fts_2$ be as defined in Definition 3.14. We have

$$\begin{aligned} \rightarrow &= \{\ t \in S_1 \uplus S_2 \times \mathcal{A}_\tau \times S_1 \uplus S_2 \mid P \models \theta'(t)\ \} \\ &= \{\ t \in S_1 \times \mathcal{A}_\tau \times S_1 \mid P \models \theta_1(t)\ \} \uplus \{\ t \in S_2 \times \mathcal{A}_\tau \times S_2 \mid P \models \theta_2(t)\ \} \\ &= \rightarrow' \end{aligned}$$

- The initial state is unchanged under projection, and hence $s_* = s_{*1} = s'_*$.

□

Appendix B

Full proofs for Chapter 4

This appendix contains the full proofs that were omitted in Chapter 4. Section B.1 of this appendix contains the omitted proofs from Section 4.1, and Section B.2 of this appendix contains the omitted proofs from Section 4.2.

B.1 Full proofs for Section 3.1

Corollary 4.1. *Let $fks_i = (S_i, AP, \theta_i, L_i, s_{*i})$, for $i \in \{1,2\}$, be two FKS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression. It holds that*

$$fks_1 \overset{\varphi}{\leftrightarrows}_f fks_2 \;\Rightarrow\; \forall P \in \mathcal{P}\colon P \models \varphi \;\Rightarrow\; fks_1|_P \leftrightarrows fks_2|_P.$$

Proof. Let $fks_i = (S_i, AP, \theta_i, L_i, s_{*i})$, for $i \in \{1,2\}$, be two FKS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression, such that $fks_1 \overset{\varphi}{\leftrightarrows}_f fks_2$. By Definition 4.2 this means $fks_1 \uplus fks_2 \models s_{*1} \overset{\varphi}{\leftrightarrows}_f s_{*2}$. By Theorem 4.1 this means that $(fks_1 \uplus fks_2)|_P \models s_{*1} \leftrightarrows s_{*2}$, and hence $fks_1|_P \uplus fks_2|_P \models s_{*1} \leftrightarrows s_{*2}$ by Lemma 3.1, for all $P \in \mathcal{P}$ such that $P \models \varphi$

By Definition 4.1 this is equivalent to $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fks_1|_P \leftrightarrows fks_2|_P$. □

Corollary 4.2. *Let $fks_i = (S_i, AP, \theta_i, L_i, s_{*i})$, for $i \in \{1,2\}$, be two FKS such that $L_1(s_{*1}) = L_2(s_{*2})$, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression. It holds that*

$$(\,\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fks_1|_P \leftrightarrows fks_2|_P\,) \;\Rightarrow\; fks_1 \overset{\varphi}{\leftrightarrows}_f fks_2.$$

Proof. Let $fks_i = (S_i, AP, \theta_i, L_i, s_{*i})$, for $i \in \{1,2\}$, be two FKS such that $L_1(s_{*1}) = L_2(s_{*2})$, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression, such that $\forall P \in \mathcal{P}\colon (\,P \models \varphi \Rightarrow fks_1|_P \leftrightarrows fks_2|_P\,)$. By Definition 4.1 this means $fks_1|_P \uplus fks_2|_P \models s_{*1} \leftrightarrows s_{*2}$, and hence $(fks_1 \uplus fks_2)|_P \models s_{*1} \leftrightarrows s_{*2}$ by Lemma 3.1, for all $P \in \mathcal{P}$ such that $P \models \varphi$.

Since $(L_1 \uplus L_2)(s_{*1}) = L_1(s_{*1}) = L_2(s_{*2}) = (L_1 \uplus L_2)(s_{*2})$, by Theorem 4.2 this implies that $fks_1 \uplus fks_2 \models s_{*1} \overset{\varphi}{\leftrightarrows}_f s_{*2}$.

By Definition 4.2 this is equivalent to $fks_1 \overset{\varphi}{\leftrightarrows}_f fks_2$. □

Theorem 4.5. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, and let $s_1, s_2 \in S$ and $\phi \in \mathbb{B}(\mathcal{F})$. It holds that*

$$fts \models s_1 \overset{\phi}{\leftrightarrows}_f s_2 \;\Rightarrow\; \forall P \in \mathcal{P}\colon (P \models \phi \;\Rightarrow\; fts|_P \models s_1 \leftrightarrows s_2).$$

Proof. Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, and let R be a feature bisimulation relation on fts such that $(s_1, \hat{\phi}, s_2) \in R$, for some $s_1, s_2 \in S$ and $\phi \in \mathbb{B}(\mathcal{F})$. Let $P \in \mathcal{P}$ be a product such that $P \models \phi$. We define the relation

$$R|_P = \{(s,t) \mid (s, \hat{\varphi}, t) \in R \wedge P \models \varphi\}.$$

We have to show that $R|_P$ is a bisimulation relation on $fts|_P$ relating s_1 and s_2. Since $(s_1, \hat{\phi}, s_2) \in R$ and $P \models \phi$, we have that $(s_1, s_2) \in R|_P$. Hence, it remains to establish that $R|_P$ is a bisimulation relation.

To show that $R|_P$ is a bisimulation relation, we first have to show that it is symmetric. Pick states $s, t \in S$ such that $(s,t) \in R|_P$. By definition of $R|_P$ this implies that $(s, \hat{\varphi}, t) \in R \wedge P \models \varphi$, for some $\varphi \in \mathbb{B}(\mathcal{F})$. Since R is symmetric we also have $(t, \hat{\varphi}, s) \in R$, and hence $(t,s) \in R|_P$, which confirms that $R|_P$ is symmetric.

Next, we have to prove that the pairs of states $(s,t) \in S \times S$ such that $(s, \hat{\varphi}, t) \in R$ and $P \models \varphi$, for some $\varphi \in \mathbb{B}(\mathcal{F})$, satisfy the transfer condition from Definition 4.4.

Suppose that $s \xrightarrow{\alpha} s'$ for some $s' \in S$ and for some $\alpha \in \mathcal{A}_\tau$. It must be shown that

- there exists $t' \in S$ such that $t \xrightarrow{\alpha} t'$ and $(s', t') \in R|_P$.

By Definition 3.15 it follows that $s \xrightarrow{\alpha|\psi} s'$ in fts, for some $\psi \in \mathbb{B}(\mathcal{F})$ such that $P \models \psi$. Since $(s, \hat{\varphi}, t) \in R$ for some $\varphi \in \mathbb{B}(\mathcal{F})$ such that $P \models \varphi$, by Definition 4.5 we have that there exist states $t_i \in S$ and feature expressions $\psi_i, \varphi_i \in \mathbb{B}(\mathcal{F})$, for $1 \leq i \leq n$, for some $n \in \mathbb{N}$, such that

$$t \xrightarrow{\alpha|\psi_i} t_i \text{ and } (s', \hat{\varphi}_i, t_i) \in R,$$

for all $1 \leq i \leq n$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}}\; \textstyle\bigvee_{1 \leq i \leq n} \psi_i \wedge \varphi_i.$$

Since $P \models \varphi \wedge \psi$ it follows that $P \models \bigvee_{1 \leq i \leq n} \psi_i \wedge \varphi_i$. We pick i such that $P \models \psi_i \wedge \varphi_i$. Since $t \xrightarrow{\alpha|\psi_i} t_i$, we have that $t \xrightarrow{\alpha} t_i$ in $fts|_P$, by Definition 3.15. Furthermore, since $(s', \hat{\varphi}_i, t_i) \in R$ and $P \models \varphi_i$, by definition of $R|_P$ we have that $(s', t') \in R|_P$.

This satisfies the transfer condition, confirming that $R|_P$ is indeed a bisimulation relation on $fts|_P$. □

Corollary 4.3. *Let $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*i})$, for $i \in \{1,2\}$, be two FTS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression. It holds that*

$$fts_1 \overset{\varphi}{\leftrightarrows}_f fts_2 \;\Rightarrow\; \forall P \in \mathcal{P}\colon (\, P \models \varphi \;\Rightarrow\; fts_1|_P \leftrightarrows fts_2|_P \,)$$

Proof. Let $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*i})$, for $i \in \{1,2\}$, be two FTS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression, such that $fts_1 \overset{\varphi}{\leftrightarrows}_f fts_2$. By Definition 4.5 this means $fts_1 \uplus fts_2 \models s_{*1} \overset{\varphi}{\leftrightarrows}_f s_{*2}$. By Theorem 4.5 this means that $(fts_1 \uplus fts_2)|_P \models s_{*1} \leftrightarrows s_{*2}$, and hence $fts_1|_P \uplus fts_2|_P \models s_{*1} \leftrightarrows s_{*2}$ by Lemma 3.2, for all $P \in \mathcal{P}$ such that $P \models \varphi$.

By Definition 4.4 this is equivalent to $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts_1|_P \leftrightarrows fts_2|_P$. □

Theorem 4.6. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, and let $s_1, s_2 \in S$ and $\phi \in \mathbb{B}(\mathcal{F})$. It holds that*

$$(\forall P \in \mathcal{P}\colon P \models \phi \;\Rightarrow\; fts|_P \models s_1 \leftrightarrows s_2) \;\Rightarrow\; fts \models s_1 \overset{\phi}{\leftrightarrows}_f s_2.$$

Proof. Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, let $s_1, s_2 \in S$ be two states, and let $\phi \in \mathbb{B}(\mathcal{F})$ be a feature expression. For all product $P \in \mathcal{P}$ such that $P \models \phi$, let $R|_P$ be a bisimulation relation on $fts|_P$ such that $(s_1, s_2) \in R|_P$. For all all products $P \in \mathcal{P}$ such that $P \not\models \phi$, let $R|_P$ be the empty bisimulation relation.

We define the relation

$$R = \{(s, \hat{\varphi}, t) \mid \forall P \in \mathcal{P} : P \models \varphi \Leftrightarrow (s, t) \in R|_P\}.$$

We have to show that R is a feature bisimulation relation on fts such that $(s_1, \hat{\phi}, s_2) \in R$. Since $(s_1, s_2) \in R|_P \Leftrightarrow P \models \phi$, for all $P \in \mathcal{P}$, it immediately follows that $(s_1, \hat{\phi}, s_2) \in R$. Hence, it remains to establish that R is a feature bisimulation relation.

To show that R is a feature bisimulation relation, we first have to show that it is symmetric. Pick states $s, t \in S$ and feature expression $\varphi \in \mathbb{B}(\mathcal{F})$ such that $(s, \hat{\varphi}, t) \in R$. By definition of R this implies that $\forall P \in \mathcal{P} : P \models \varphi \Leftrightarrow (s, t) \in R|_P$. Since $R|_P$ is symmetric, for each $P \in \mathcal{P}$, we also have $\forall P \in \mathcal{P} : P \models \varphi \Leftrightarrow (t, s) \in R|_P$, and hence $(t, \hat{\varphi}, s) \in R$, which confirms that R is symmetric.

Next, we have to prove that the tuples $(s, \hat{\varphi}, t) \in S \times \mathbb{B}(\mathcal{F}) \times S$ such that $\forall P \in \mathcal{P} : P \models \varphi \Leftrightarrow (s, t) \in R|_P$, satisfy the transfer condition from Definition 4.5.

Suppose that $s \xrightarrow{\alpha|\psi} s'$, for some $s' \in S$, for some $\alpha \in \mathcal{A}_\tau$, and for some $\psi \in \mathbb{B}(\mathcal{F})$. It must be shown that there exist states $t_i \in S$ and feature expressions $\psi_i, \varphi_i \in \mathbb{B}(\mathcal{F})$, for $1 \leq i \leq n$, for some $n \in \mathbb{N}$, such that

$$t \xrightarrow{\alpha|\psi_i} t_i \text{ and } (s', \hat{\varphi}_i, t_i) \in R,$$

for all $1 \leq i \leq n$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}}\; \bigvee_{1 \leq i \leq n} \psi_i \wedge \varphi_i.$$

By the definition of R it follows that for each $P \in \mathcal{P}$ such that $P \models \varphi$, we have that $(s, t) \in R|_P$. For each $P \in \mathcal{P}$ such that $P \models \psi$, it is the case that $s \xrightarrow{\alpha} s'$ in $fts|_P$, by Definition 3.15. Hence, for each $P \in \mathcal{P}$ such that $P \models \varphi \wedge \psi$, by definition of bisimilarity it is the case that $t \xrightarrow{\alpha} t_P$ in $fts|_P$, for some $t_P \in S$, such that $(s', t_P) \in R|_P$. By definition of projection, this means that $t \xrightarrow{\alpha|\psi_P} t_P$ in fts, for some $\psi_P \in \mathbb{B}(\mathcal{F})$ such that $P \models \psi_P$. Furthermore, by definition of R it follows that $(s', \hat{\varphi}_P, t_P) \in R$, for some $\varphi_P \in \mathbb{B}(\mathcal{F})$ such that $P \models \varphi_P$. Let $\mathcal{P}' = \{P \in \mathcal{P} \mid P \models \varphi \wedge \psi\}$ be the set of all products satisfying $\varphi \wedge \psi$. We conclude that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}}\; \bigvee_{P' \in \mathcal{P}'} \psi_{P'} \wedge \varphi_{P'}$$

Hence this satisfies the transfer condition for feature bisimilarity, proving that relation R is indeed a feature bisimulation relation on fts. □

Corollary 4.4. *Let $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*i})$, for $i \in \{1, 2\}$, be two FTS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression. It holds that*

$$(\forall P \in \mathcal{P}\colon P \models \varphi \;\Rightarrow\; fts_1|_P \leftrightarrows fts_2|_P) \;\Rightarrow\; fts_1 \overset{\varphi}{\leftrightarrows}_f fts_2.$$

Proof. Let $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*i})$, for $i \in \{1,2\}$, be two FTS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression, such that $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts_1|_P \leftrightarrows fts_2|_P$. By Definition 4.4 this means $fts_1|_P \uplus fts_2|_P \models s_{*1} \leftrightarrows s_{*2}$, and hence $(fts_1 \uplus fts_2)|_P \models s_{*1} \leftrightarrows s_{*2}$, by Lemma 3.2, for all $P \in \mathcal{P}$ such that $P \models \varphi$. By Theorem 4.6 this implies that $fts_1 \uplus fts_2 \models s_{*1} \overset{\varphi}{\leftrightarrows}_f s_{*2}$.

By Definition 4.5 this is equivalent to $fts_1 \overset{\varphi}{\leftrightarrows}_f fts_2$.

□

Theorem 4.7. *Feature bisimilarity ($\overset{\varphi}{\leftrightarrows}_f$) for FTS for given $\varphi \in \mathbb{B}(\mathcal{F})$ is an equivalence relation.*

Proof. We have to show that feature bisimilarity is reflexive, symmetric and transitive, for some feature expression $\varphi \in \mathbb{B}(\mathcal{F})$. Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS such that $s, t, u \in S$.

1. We prove reflexivity using s. By reflexivity of bisimilarity for LTS (Lemma 4.3), we have $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts \models s \leftrightarrows s$.Hence by Theorem 4.6 it follows that $fts \models s \overset{\varphi}{\leftrightarrows}_f s$. Hence $\overset{\varphi}{\leftrightarrows}_f$ is reflexive.

2. Suppose that $fts \models s \overset{\varphi}{\leftrightarrows}_f t$. By Theorem 4.5 it follows that $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts|_P \models s \leftrightarrows t$. By symmetry of bisimilarity for LTS (Lemma 4.3), we have $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts|_P \models t \leftrightarrows s$. By Theorem 4.6 it follows that $fts \models t \overset{\varphi}{\leftrightarrows}_f s$. Hence $\overset{\varphi}{\leftrightarrows}_f$ is symmetric.

3. Suppose that $fts \models s \overset{\varphi}{\leftrightarrows}_f t$ and $fts \models t \overset{\varphi}{\leftrightarrows}_f u$. By Theorem 4.5 it follows that $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts|_P \models s \leftrightarrows t \wedge fts|_P \models t \leftrightarrows u$. By transitivity of bisimilarity for LTS (Lemma 4.3), we have $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts|_P \models s \leftrightarrows u$. By Theorem 4.6 it follows that $fts \models s \overset{\varphi}{\leftrightarrows}_f u$. Hence $\overset{\varphi}{\leftrightarrows}_f$ is transitive.

□

Lemma 4.4. *Let fts be an FTS with states s, t and u such that $fts \models s \overset{\varphi}{\leftrightarrows}_f t$ and $fts \models t \overset{\psi}{\leftrightarrows}_f u$, for some $\varphi, \psi \in \mathbb{B}(\mathcal{F})$. Then $fts \models s \overset{\varphi \wedge \psi}{\longleftrightarrow}_f u$.*

Proof. Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS with such that $s, t, u \in S$, and such that $fts \models s \overset{\varphi}{\leftrightarrows}_f t$ and $fts \models t \overset{\psi}{\leftrightarrows}_f u$. By Theorem 4.5 it follows that $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts|_P \models s \leftrightarrows t$ and $\forall P \in \mathcal{P}\colon P \models \psi \Rightarrow fts|_P \models t \leftrightarrows u$. By transitivity of bisimilarity for LTS (Lemma 4.3), we have $\forall P \in \mathcal{P}\colon P \models \varphi \wedge \psi \Rightarrow fts|_P \models s \leftrightarrows u$. By Theorem 4.6 it follows that $fts \models s \overset{\varphi \wedge \psi}{\longleftrightarrow}_f u$. □

Theorem 4.8. *A property is preserved by bisimilarity for LTS if and only if this property is preserved by feature bisimilarity for FTS.*

Proof. Let fts_1 and fts_2 be FTS such that $fts_1 \leftrightarrows_f fts_2$, and let ϕ be a property that is preserved by bisimilarity for LTS. We have to show that $fts_1 \models \phi \Leftrightarrow fts_2 \models \phi$.

Assume $fts_1 \models \phi$. By Definition 3.16 this is equivalent to $\forall P \in \mathcal{P}\colon fts_1|_P \models \phi$. Since $fts_1 \leftrightarrows_f fts_2$, by Corollary 4.3 we have $\forall P \in \mathcal{P}\colon fts_1|_P \leftrightarrows fts_2|_P$. Since ϕ is preserved by $\leftrightarrows$, it follows that $\forall P \in \mathcal{P}\colon fts_2|_P \models \phi$, which is equivalent to $fts_2 \models \phi$, by Definition 3.16.

The proof for the implication in the other direction is symmetric.

□

B.2 Full proofs for Section 3.2

Theorem 4.9. *Let fks* $= (S, AP, \theta, L, s_*)$ *be an FKS, and let* $s_1, s_2 \in S$ *and* $\phi \in \mathbb{B}(\mathcal{F})$. *It holds that*

$$\mathit{fks} \models s_1 \overset{\phi}{\approx}_{\mathit{dbsf}} s_2 \;\Rightarrow\; \forall P \in \mathcal{P}\colon (P \models \phi \;\Rightarrow\; \mathit{fks}|_P \models s_1 \approx_{\mathit{dbs}} s_2).$$

Proof. Let *fks* $= (S, AP, \theta, L, s_*)$ be an FKS, and let R be a divergence-blind stuttering feature bisimulation relation on *fks* such that $(s_1, \hat{\phi}, s_2) \in R$, for some $s_1, s_2 \in S$ and $\phi \in \mathbb{B}(\mathcal{F})$. Let $P \in \mathcal{P}$ be a product such that $P \models \phi$. We define the relation

$$R|_P = \{(s,t) \mid (s, \hat{\varphi}, t) \in R \wedge P \models \varphi\}.$$

We have to show that $R|_P$ is a divergence-blind stuttering bisimulation relation on the KS $\mathit{fks}|_P = (S, AP, \rightarrow, L, s_*)$ relating s_1 and s_2. Since $(s_1, \hat{\phi}, s_2) \in R$ and $P \models \phi$, we have that $s_1 \, R|_P \, s_2$. Hence, it remains to establish that $R|_P$ is a divergence-blind stuttering bisimulation relation.

To show that $R|_P$ is a divergence-blind stuttering bisimulation relation, we first have to show that it is symmetric. Pick states $s, t \in S$ such that $(s,t) \in R|_P$. By definition of $R|_P$ this implies that $(s, \hat{\varphi}, t) \in R \wedge P \models \varphi$, for some $\varphi \in \mathbb{B}(\mathcal{F})$. Since R is symmetric we also have $(t, \hat{\varphi}, s) \in R$, and hence $(t,s) \in R|_P$, which confirms that $R|_P$ is symmetric.

Next, we have to prove that the pairs of states $(s,t) \in S \times S$ such that $(s, \hat{\varphi}, t) \in R$ and $P \models \varphi$, for some $\varphi \in \mathbb{B}(\mathcal{F})$, satisfy the conditions from Definition 4.7.

Since R is a divergence-blind stuttering feature bisimulation relation on *fks* and $(s, \hat{\varphi}, t) \in R$, by Definition 4.9 we have $L(s) = L(t)$, and hence the first condition is satisfied.

Suppose that $s \rightarrow s'$ for some $s' \in S$. It must be shown that

- there exist states $t', t'' \in S$ such that $t \longrightarrow\!\!\!\!\!\rightarrow t' \dashrightarrow t''$ and $(s,t') \in R|_P$ and $(s',t'') \in R|_P$.

By Definition 3.7 it follows that $s \xrightarrow{\psi} s'$ in *fks*, for some $\psi \in \mathbb{B}(\mathcal{F})$ such that $P \models \psi$. Since $(s, \hat{\varphi}, t) \in R$ for some $\varphi \in \mathbb{B}(\mathcal{F})$ such that $P \models \varphi$, by Definition 4.9 we have can find states $t_i', t_i'' \in S$ and feature expressions $\Psi_i, \psi_i, \varphi_i, \varphi_i' \in \mathbb{B}(\mathcal{F})$, for $1 \leq i \leq n$, for some $n \in \mathbb{N}$, such that

$$t \xrightarrow{\Psi_i}\!\!\!\!\!\rightarrow t_i' \overset{\psi_i}{\dashrightarrow} t_i \text{ and } (s, \hat{\varphi}_i, t_i') \in R \text{ and } (s', \hat{\varphi}_i', t_i'') \in R,$$

for all $1 \leq i \leq n$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}}\; \bigvee_{1 \leq i \leq n} \Psi_i \wedge \psi_i \wedge \varphi_i \wedge \varphi_i'.$$

Since $P \models \varphi \wedge \psi$ it follows that $P \models \bigvee_{1 \leq i \leq n} \Psi_i \wedge \psi_i \wedge \varphi_i \wedge \varphi_i'$. We pick i such that $P \models \Psi_i \wedge \psi_i \wedge \varphi_i \wedge \varphi_i'$. Since $t \xrightarrow{\Psi}\!\!\!\!\!\rightarrow t_i' \overset{\psi_i}{\dashrightarrow} t_i''$, we have that $t \longrightarrow\!\!\!\!\!\rightarrow t_i' \dashrightarrow t_i''$ in $\mathit{fks}|_P$, by Definition 3.7. Furthermore, since $(s, \hat{\varphi}_i, t_i') \in R$ and $(s, \hat{\varphi}_i', t_i'') \in R$ and $P \models \varphi_i \wedge \varphi_i'$, by definition of $R|_P$ we have that $(s, t_i') \in R|_P$ and $(s', t_i'') \in R|_P$.

This satisfies the second condition, confirming that $R|_P$ is indeed a divergence-blind stuttering bisimulation relation on $\mathit{fks}|_P$. □

Corollary 4.5. *Let $fks_i = (S_i, AP, \theta_i, L_i, s_{*i})$, for $i \in \{1,2\}$, be two FKS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression. It holds that*

$$fks_1 \overset{\varphi}{\approx}_{dbsf} fks_2 \;\Rightarrow\; (\forall P \in \mathcal{P}\colon P \models \varphi \;\Rightarrow\; fks_1|_P \approx_{dbs} fks_2|_P\,).$$

Proof. Let $fks_i = (S_i, AP, \theta_i, L_i, s_{*i})$, for $i \in \{1,2\}$, be two FKS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression, such that $fks_1 \overset{\varphi}{\approx}_{dbsf} fks_2$. By Definition 4.9 this means $fks_1 \uplus fks_2 \models s_{*1} \overset{\varphi}{\approx}_{dbsf} s_{*2}$. By Theorem 4.9 this means that $(fks_1 \uplus fks_2)|_P \models s_{*1} \approx_{dbs} s_{*2}$, and hence $fks_1|_P \uplus fks_2|_P \models s_{*1} \approx_{dbs} s_{*2}$ by Lemma 3.1, for all $P \in \mathcal{P}$ such that $P \models \varphi$

By Definition 4.7 this is equivalent to $\forall P \in \mathcal{P}\colon P \models \varphi \;\Rightarrow\; fks_1|_P \approx_{dbs} fks_2|_P$. □

Theorem 4.10. *Let $fks = (S, AP, \theta, L, s_*)$ be an FKS, and let $s_1, s_2 \in S$ such that $L(s_1) = L(s_2)$, and let $\phi \in \mathbb{B}(\mathcal{F})$. It holds that*

$$(\forall P \in \mathcal{P}\colon P \models \phi \;\Rightarrow\; fks|_P \models s_1 \approx_{dbs} s_2) \;\Rightarrow\; fks \models s_1 \overset{\phi}{\approx}_{dbsf} s_2.$$

Proof. Let $fks = (S, AP, \theta, L, s_*)$ be an FKS, let $s_1, s_2 \in S$ be two states such that $L(s_1) = L(s_2)$, and let $\phi \in \mathbb{B}(\mathcal{F})$ be a feature expression. For all products $P \in \mathcal{P}$ such that $P \models \phi$, let $R|_P$ be a divergence-blind stuttering bisimulation relation on $fks|_P$, such that $(s_1, s_2) \in R|_P$. For all products $P \in \mathcal{P}$ such that $P \not\models \phi$, let $R|_P = \emptyset$ be the empty divergence-blind stuttering bisimulation relation.

We define the relation

$$R = \{\, (s, \hat{\varphi}, t) \mid L(s) = L(t) \wedge \forall P \in \mathcal{P} : P \models \varphi \Leftrightarrow (s,t) \in R|_P \,\}.$$

We have to show that R is a divergence-blind stuttering feature bisimulation relation on fks such that $(s_1, \hat{\phi}, s_2) \in R$. Since $(s_1, s_2) \in R|_P \Leftrightarrow P \models \phi$, for all $P \in \mathcal{P}$, and $L(s_1) = L(s_2)$, it immediately follows that $(s_1, \hat{\phi}, s_2) \in R$. Hence, it remains to establish that R is a divergence-blind stuttering feature bisimulation relation.

To show that R is a divergence-blind stuttering feature bisimulation relation, we first have to show that it is symmetric. Pick states $s, t \in S$ and feature expression $\varphi \in \mathbb{B}(\mathcal{F})$ such that $(s, \hat{\varphi}, t) \in R$. By definition of R this implies that $L(s) = L(t) \wedge \forall P \in \mathcal{P} : (\, P \models \varphi \Leftrightarrow (s,t) \in R|_P\,)$. Since $R|_P$ is symmetric, for each $P \in \mathcal{P}$, we also have $L(t) = L(s) \wedge \forall P \in \mathcal{P} : (\, P \models \varphi \Leftrightarrow (t,s) \in R|_P\,)$, and hence $(t, \hat{\varphi}, s) \in R$, which confirms that R is symmetric.

Next, we have to prove that tuples $(s, \hat{\varphi}, t) \in S \times \mathbb{B}(\mathcal{F}) \times S$ such that $L(s) = L(t)$ and $\forall P \in \mathcal{P} : (\, P \models \varphi \Leftrightarrow (s,t) \in R|_P\,)$, satisfy the conditions from Definition 4.9.

By construction we have that $L(s) = L(t)$. Hence the first condition is satisfied.

Suppose that $s \xrightarrow{\psi} s'$, for some $s' \in S$ and for some $\psi \in \mathbb{B}(\mathcal{F})$. It must be shown that there exist states $t_i', t_i'' \in S$ and feature expressions $\Psi_i, \psi_i, \varphi_i, \varphi_i' \in \mathbb{B}(\mathcal{F})$, for $1 \le i \le n$, for some $n \in \mathbb{N}$, such that

$$t \xrightarrow{\Psi_i}\!\!\!\!\rightarrow t_i' \overset{\psi_i}{\dashrightarrow} t_i' \text{ and } (s, \hat{\varphi}_i, t_i') \in R \text{ and } (s', \hat{\varphi}_i', t_i'') \in R,$$

for all $1 \le i \le n$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}}\; \bigvee_{1 \le i \le n} \Psi_i \wedge \psi_i \wedge \varphi_i \wedge \varphi_i'.$$

By definition of R it follows that for each $P \in \mathcal{P}$ such that $P \models \varphi$, we have that $(s,t) \in R|_P$. For each $P \in \mathcal{P}$ such that $P \models \psi$, it is the case that $s \to s'$ in $fks|_P$, by Definition 3.7. Hence, for each $P \in \mathcal{P}$ such that $P \models \varphi \wedge \psi$, by definition of divergence-blind stuttering equivalence for $R|_P$ it is the case that $t \longrightarrow\!\!\!\!\!\rightarrow t'_P \dashrightarrow t''_P$ in $fks|_P$, for some $t'_P, t''_P \in S$, such that $(s,t'_P) \in R|_P$ and $(s',t''_P) \in R|_P$. By definition of projection, this means that $t \xrightarrow{\Psi_P}\!\!\!\!\!\rightarrow t'_P \overset{\psi_P}{\dashrightarrow} t''_P$ in fks, for some $\Psi_P, \psi_P \in \mathbb{B}(\mathcal{F})$ such that $P \models \Psi_P \wedge \psi_P$. Furthermore, by definition of R it follows that $(s, \hat{\varphi}_P, t'_P) \in R$ and $(s', \hat{\varphi}'_P, t''_P) \in R$, for some $\varphi_P, \varphi'_P \in \mathbb{B}(\mathcal{F})$ such that $P \models \varphi_P \wedge \varphi'_P$. Let $\mathcal{P}' = \{P \in \mathcal{P} \mid P \models \varphi \wedge \psi\}$ be the set of all products satisfying $\varphi \wedge \psi$. We conclude that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}}\; \bigvee_{P' \in \mathcal{P}'} \Psi_{P'} \wedge \psi_{P'} \wedge \varphi_{P'} \wedge \varphi'_{P'},$$

which this satisfies the second condition for divergence-blind stuttering feature bisimulation, proving that relation R is indeed a divergence-blind stuttering feature bisimulation relation on fks. □

Corollary 4.6. *Let $fks_i = (S_i,\ AP,\ \theta_i,\ L_i,\ s_{*i})$, for $i \in \{1,2\}$, be two FKS such that $L_1(s_{*1}) = L_2(s_{*2})$, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression. It holds that*

$$(\,\forall P \in \mathcal{P}\colon P \models \varphi \;\Rightarrow\; fks_1|_P \approx_{dbs} fks_2|_P\,) \;\Rightarrow\; fks_1 \overset{\varphi}{\approx}_{dbsf} fks_2.$$

Proof. Let $fks_i = (S_i,\ AP,\ \theta_i,\ L_i,\ s_{*i})$, for $i \in \{1,2\}$, be two FKS such that $L_1(s_{*1}) = L_2(s_{*2})$, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression, such that $\forall P \in \mathcal{P}\colon (\,P \models \varphi \;\Rightarrow\; fks_1|_P \approx_{dbs} fks_2|_P\,)$. By Definition 4.7 this means $fks_1|_P \uplus fks_2|_P \models s_{*1} \approx_{dbs} s_{*2}$, and hence $(fks_1 \uplus fks_2)|_P \models s_{*1} \approx_{dbs} s_{*2}$ by Lemma 3.1, for all $P \in \mathcal{P}$ such that $P \models \varphi$.

Since $(L_1 \uplus L_2)(s_{*1}) = L_1(s_{*1}) = L_2(s_{*2}) = (L_1 \uplus L_2)(s_{*2})$, by Theorem 4.10 this implies that $fks_1 \uplus fks_2 \models s_{*1} \overset{\varphi}{\approx}_{dbsf} s_{*2}$.

By Definition 4.9 this is equivalent to $fks_1 \overset{\varphi}{\approx}_{dbsf} fks_2$. □

Theorem 4.11. *Divergence-blind stuttering feature equivalence ($\overset{\varphi}{\approx}_{dbsf}$) for given $\varphi \in \mathbb{B}(\mathcal{F})$ is an equivalence relation.*

Proof. We have to show that divergence-blind stuttering feature equivalence is reflexive, symmetric and transitive, for some feature expression $\varphi \in \mathbb{B}(\mathcal{F})$. Let $fks = (S,\ AP,\ \theta,\ L,\ s_*)$ be an FKS such that $s,t,u \in S$.

1. We prove reflexivity using s. By reflexivity of divergence-blind stuttering equivalence for KS (Lemma 4.5), we have $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fks \models s \approx_{dbs} s$. Obviously $L(s) = L(s)$, and hence by Theorem 4.10 it follows that $fks \models s \overset{\varphi}{\approx}_{dbsf} s$. Hence $\overset{\varphi}{\approx}_{dbsf}$ is reflexive.

2. Suppose that $fks \models s \overset{\varphi}{\approx}_{dbsf} t$, from which it follows that $L(s) = L(t)$. By Theorem 4.9 it follows that $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fks|_P \models s \approx_{dbs} t$. By symmetry of divergence-blind stuttering equivalence for KS (Lemma 4.5), we have $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fks|_P \models t \approx_{dbs} s$. Since $L(t) = L(s)$, by Theorem 4.10 it follows that $fks \models t \overset{\varphi}{\approx}_{dbsf} s$. Hence $\overset{\varphi}{\approx}_{dbsf}$ is symmetric.

3. Suppose that $fks \models s \overset{\varphi}{\approx}_{dbsf} t$ and $fks \models t \overset{\varphi}{\approx}_{dbsf} u$, from which it follows that $L(s) = L(t) = L(u)$. By Theorem 4.9 it follows that $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fks|_P \models s \approx_{dbs} t \wedge fks|_P \models t \approx_{dbs} u$. By transitivity of divergence-blind stuttering equivalence for KS (Lemma 4.5), we have $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fks|_P \models s \approx_{dbs} u$. Since $L(s) = L(u)$, by Theorem 4.10 it follows that $fks \models s \overset{\varphi}{\approx}_{dbsf} u$. Hence $\overset{\varphi}{\approx}_{dbsf}$ is transitive.

□

Lemma 4.6. *Let fks be an FKS with states s, t and u such that $fks \models s \overset{\varphi}{\approx}_{dbsf} t$ and $fks \models t \overset{\psi}{\approx}_{dbsf} u$, for some $\varphi, \psi \in \mathbb{B}(\mathcal{F})$. Then $fks \models s \overset{\varphi \wedge \psi}{\approx}_{dbsf} u$.*

Proof. Let $fks = (S, AP, \theta, L, s_*)$ be an FKS with such that $s, t, u \in S$, and such that $fks \models s \overset{\varphi}{\approx}_{dbsf} t$ and $fks \models t \overset{\psi}{\approx}_{dbsf} u$, from which it follows that $L(s) = L(t) = L(u)$. By Theorem 4.9 it follows that $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fks|_P \models s \approx_{dbs} t$ and $\forall P \in \mathcal{P}\colon P \models \psi \Rightarrow fks|_P \models t \approx_{dbs} u$. By transitivity of divergence-blind stuttering equivalence for KS (Lemma 4.5), we have $\forall P \in \mathcal{P}\colon P \models \varphi \wedge \psi \Rightarrow fks|_P \models s \approx_{dbs} u$. Since $L(s) = L(u)$, by Theorem 4.10 it follows that $fks \models s \overset{\varphi \wedge \psi}{\approx}_{dbsf} u$. □

Theorem 4.12. *A property is preserved by divergence-blind stuttering equivalence for KS if and only if this property is preserved by divergence-blind stuttering feature equivalence for FKS.*

Proof. Let fks_1 and fks_2 be FKS such that $fks_1 \approx_{dbsf} fks_2$, and let ϕ be a property that is preserved by divergence-blind stuttering equivalence for KS. We have to show that $fks_1 \models \phi \Leftrightarrow fks_2 \models \phi$.

Assume $fks_1 \models \phi$. By Definition 3.8 this is equivalent to $\forall P \in \mathcal{P}\colon fks_1|_P \models \phi$. Since $fks_1 \approx_{dbsf} fks_2$, by Corollary 4.5 we have $\forall P \in \mathcal{P}\colon fks_1|_P \approx_{dbs} fks_2|_P$. Since ϕ is preserved by $\leftrightarrows$, it follows that $\forall P \in \mathcal{P}\colon fks_2|_P \models \phi$, which is equivalent to $fks_2 \models \phi$, by Definition 3.8.

The proof for the implication in the other direction is similar. □

Theorem 4.13. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, and let $s_1, s_2 \in S$ and $\phi \in \mathbb{B}(\mathcal{F})$. It holds that*

$$fts \models s_1 \overset{\phi}{\leftrightarrows}_{bf} s_2 \;\Rightarrow\; \forall P \in \mathcal{P}\colon (P \models \phi \;\Rightarrow\; fts|_P \models s_1 \leftrightarrows_b s_2).$$

Proof. Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, and let R be a branching feature bisimulation relation on fts such that $(s_1, \hat{\phi}, s_2) \in R$, for some $s_1, s_2 \in S$ and $\phi \in \mathbb{B}(\mathcal{F})$. Let $P \in \mathcal{P}$ be a product such that $P \models \phi$. We define the relation

$$R|_P = \{(s, t) \mid (s, \hat{\varphi}, t) \in R \wedge P \models \varphi\}.$$

We have to show that $R|_P$ is a branching bisimulation relation on the LTS $fts|_P = (S, \mathcal{A}, \rightarrow, s_*)$ relating s_1 and s_2. Since $(s_1, \hat{\phi}, s_2) \in R$ and $P \models \phi$, we have that $s_1 \, R|_P \, s_2$. Hence, it remains to establish that $R|_P$ is a branching bisimulation relation.

To show that $R|_P$ is a branching bisimulation relation, we first have to show that it is symmetric. Pick states $s, t \in S$ such that $(s, t) \in R|_P$. By definition of $R|_P$ this implies that $(s, \hat{\varphi}, t) \in R \wedge P \models \varphi$, for some $\varphi \in \mathbb{B}(\mathcal{F})$. Since R is symmetric we also have $(t, \hat{\varphi}, s) \in R$, and hence $(t, s) \in R|_P$, which confirms that $R|_P$ is symmetric.

Next, we have to prove that the pairs of states $(s,t) \in S \times S$ such that $(s,\hat{\varphi},t) \in R$ and $P \models \varphi$, for some $\varphi \in \mathbb{B}(\mathcal{F})$, satisfy the transfer condition from Definition 4.11.

Suppose that $s \xrightarrow{\alpha} s'$ in $fts|_P$, for some $s' \in S$ and for some $\alpha \in \mathcal{A}_\tau$. It must be shown that

- there exist states $t', t'' \in S$ such that $t \longrightarrow\!\!\!\!\!\rightarrow t' \xrightarrow{(\alpha)} t''$ and $(s,t') \in R|_P$ and $(s',t'') \in R|_P$.

By Definition 3.15 it follows that $s \xrightarrow{\alpha|\psi} s'$ in fts, for some $\psi \in \mathbb{B}(\mathcal{F})$ such that $P \models \psi$. Since $(s,\hat{\varphi},t) \in R$ for some $\varphi \in \mathbb{B}(\mathcal{F})$ such that $P \models \varphi$, by Definition 4.13 we have can find states $t'_0, t''_i \in S$ and feature expressions $\Psi_i, \psi_i, \varphi_i, \varphi'_i \in \mathbb{B}(\mathcal{F})$, for $1 \leq i \leq n$, for some $n \in \mathbb{N}$, such that

$$t \xrightarrow{\Psi_i}\!\!\!\!\!\rightarrow t'_i \xrightarrow{(\alpha|\psi_i)} t_i \text{ and } (s,\hat{\varphi}_i,t'_i) \in R \text{ and } (s',\hat{\varphi}'_i,t''_i) \in R,$$

for all $1 \leq i \leq n$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}}\; \bigvee_{1 \leq i \leq n} \Psi_i \wedge \psi_i \wedge \varphi_i \wedge \varphi'_i.$$

Since $P \models \varphi \wedge \psi$ it follows that $P \models \bigvee_{1 \leq i \leq n} \Psi_i \wedge \psi_i \wedge \varphi_i \wedge \varphi'_i$. We pick i such that $P \models \Psi_i \wedge \psi_i \wedge \varphi_i \wedge \varphi'_i$. Since $t \xrightarrow{\Psi}\!\!\!\!\!\rightarrow t'_i \xrightarrow{(\alpha|\psi_i)} t''_i$, we have that $t \longrightarrow\!\!\!\!\!\rightarrow t'_i \xrightarrow{(\alpha)} t''_i$ in $fts|_P$, by Definition 3.15. Furthermore, since $(s,\hat{\varphi}_i,t'_i) \in R$ and $(s,\hat{\varphi}'_i,t''_i) \in R$ and $P \models \varphi_i \wedge \varphi'_i$, by definition of $R|_P$ we have that $(s,t'_i) \in R|_P$ and $(s',t''_i) \in R|_P$.

This satisfies the transfer condition, confirming that $R|_P$ is indeed a branching bisimulation relation on $fts|_P$. □

Corollary 4.7. *Let $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*i})$, for $i \in \{1,2\}$, be two FTS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression. It holds that*

$$fts_1 \overset{\varphi}{\leftrightarrows}_{bf} fts_2 \;\Rightarrow\; (\forall P \in \mathcal{P}\colon P \models \varphi \;\Rightarrow\; fts_1|_P \leftrightarrows_b fts_2|_P\,).$$

Proof. Let $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*i})$, for $i \in \{1,2\}$, be two FTS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression, such that $fts_1 \overset{\varphi}{\leftrightarrows}_{bf} fts_2$. By Definition 4.13 this means $fts_1 \uplus fts_2 \models s_{*1} \overset{\varphi}{\leftrightarrows}_{bf} s_{*2}$. By Theorem 4.13 this means that $(fts_1 \uplus fts_2)|_P \models s_{*1} \leftrightarrows_{\mathrm{b}} s_{*2}$, and hence $fts_1|_P \uplus fts_2|_P \models s_{*1} \leftrightarrows_{\mathrm{b}} s_{*2}$ by Lemma 3.2, for all $P \in \mathcal{P}$ such that $P \models \varphi$

By Definition 4.11 this is equivalent to $\forall P \in \mathcal{P}\colon P \models \varphi \;\Rightarrow\; fts_1|_P \leftrightarrows_{\mathrm{b}} fts_2|_P$. □

Theorem 4.14. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, let $s_1, s_2 \in S$, and let $\phi \in \mathbb{B}(\mathcal{F})$. It holds that*

$$(\forall P \in \mathcal{P}\colon P \models \phi \;\Rightarrow\; fts|_P \models s_1 \leftrightarrows_b s_2) \;\Rightarrow\; fts \models s_1 \overset{\phi}{\leftrightarrows}_{bf} s_2.$$

Proof. Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, let $s_1, s_2 \in S$ be two states, and let $\phi \in \mathbb{B}(\mathcal{F})$ be a feature expression. For all products $P \in \mathcal{P}$ such that $P \models \phi$, let $R|_P$ be a branching bisimulation relation on $fts|_P$, such that $(s_1,s_2) \in R|_P$. For all products $P \in \mathcal{P}$ such that $P \not\models \phi$, let $R|_P = \emptyset$ be the empty branching bisimulation relation.

We define the relation

$$R = \{\, (s,\hat{\varphi},t) \mid \forall P \in \mathcal{P} : P \models \varphi \Leftrightarrow (s,t) \in R|_P \,\}.$$

We have to show that R is a branching feature bisimulation relation on *fts* such that $(s_1, \hat{\phi}, s_2) \in R$. Since $(s_1, s_2) \in R|_P \Leftrightarrow P \models \phi$, for all $P \in \mathcal{P}$, it immediately follows that $(s_1, \hat{\phi}, s_2) \in R$. Hence, it remains to establish that R is a branching feature bisimulation relation.

To show that R is a branching feature bisimulation relation, we first have to show that it is symmetric. Pick states $s, t \in S$ and feature expression $\varphi \in \mathbb{B}(\mathcal{F})$ such that $(s, \hat{\varphi}, t) \in R$. By definition of R this implies that $\forall P \in \mathcal{P} : (\, P \models \varphi \Leftrightarrow (s,t) \in R|_P \,)$. Since $R|_P$ is symmetric, for each $P \in \mathcal{P}$, we also have $\forall P \in \mathcal{P} : (\, P \models \varphi \Leftrightarrow (t,s) \in R|_P \,)$, and hence $(t, \hat{\varphi}, s) \in R$, which confirms that R is symmetric.

Next, we have to prove that tuples $(s, \hat{\varphi}, t) \in S \times \mathbb{B}(\mathcal{F}) \times S$ such that $\forall P \in \mathcal{P} : (\, P \models \varphi \Leftrightarrow (s,t) \in R|_P \,)$, satisfy the transfer condition from Definition 4.13.

Suppose that $s \xrightarrow{\alpha|\psi} s'$, for some $s' \in S$, $\alpha \in \mathcal{A}_\tau$ and $\psi \in \mathbb{B}(\mathcal{F})$. It must be shown that there exist states $t'_i, t''_i \in S$ and feature expressions $\Psi_i, \psi_i, \varphi_i, \varphi'_i \in \mathbb{B}(\mathcal{F})$, for $1 \leq i \leq n$, for some $n \in \mathbb{N}$, such that

$$t \xrightarrow{\Psi_i} \!\!\!\!\rightarrow\, t'_i \xrightarrow{(\alpha|\psi_i)} t'_i \text{ and } (s, \hat{\varphi}_i, t'_i) \in R \text{ and } (s', \hat{\varphi}'_i, t''_i) \in R,$$

for all $1 \leq i \leq n$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}}\; \bigvee_{1 \leq i \leq n} \Psi_i \wedge \psi_i \wedge \varphi_i \wedge \varphi'_i.$$

By definition of R it follows that for each $P \in \mathcal{P}$ such that $P \models \varphi$, we have that $(s,t) \in R|_P$. For each $P \in \mathcal{P}$ such that $P \models \psi$, it is the case that $s \xrightarrow{\alpha} s'$ in $fks|_P$, by Definition 3.15. Hence, for each $P \in \mathcal{P}$ such that $P \models \varphi \wedge \psi$, by definition of branching bisimulation for $R|_P$ it is the case that $t \longrightarrow\!\!\!\!\rightarrow\, t'_P \xrightarrow{(\alpha)} t''_P$ in $fts|_P$, for some $t'_P, t''_P \in S$, such that $(s, t'_P) \in R|_P$ and $(s', t''_P) \in R|_P$. By definition of projection, this means that $t \xrightarrow{\Psi_P} \!\!\!\!\rightarrow\, t'_P \xrightarrow{(\alpha|\psi_P)} t''_P$ in *fts*, for some $\Psi_P, \psi_P \in \mathbb{B}(\mathcal{F})$ such that $P \models \Psi_P \wedge \psi_P$. Furthermore, by definition of R it follows that $(s, \hat{\varphi}_P, t'_P) \in R$ and $(s', \hat{\varphi}'_P, t''_P) \in R$, for some $\varphi_P, \varphi'_P \in \mathbb{B}(\mathcal{F})$ such that $P \models \varphi_P \wedge \varphi'_P$. Let $\mathcal{P}' = \{P \in \mathcal{P} \mid P \models \varphi \wedge \psi\}$ be the set of all products satisfying $\varphi \wedge \psi$. We conclude that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}}\; \bigvee_{P' \in \mathcal{P}'} \Psi_{P'} \wedge \psi_{P'} \wedge \varphi_{P'} \wedge \varphi'_{P'},$$

which this satisfies the transfer condition of branching feature bisimulation, proving that relation R is indeed a branching feature bisimulation relation on *fts*. □

Corollary 4.8. *Let $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*i})$, for $i \in \{1,2\}$, be two FTS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression. It holds that*

$$(\, \forall P \in \mathcal{P} \colon P \models \varphi \;\Rightarrow\; fts_1|_P \leftrightarrows_b fts_2|_P \,) \;\Rightarrow\; fts_1 \overset{\varphi}{\leftrightarrows}_{bf} fts_2.$$

Proof. Let $fts_i = (S_i, \mathcal{A}, \theta_i, s_{*i})$, for $i \in \{1,2\}$, be two FTS, and let $\varphi \in \mathbb{B}(\mathcal{F})$ be a feature expression, such that $\forall P \in \mathcal{P} \colon (\, P \models \varphi \;\Rightarrow\; fts_1|_P \leftrightarrows_{\mathrm{b}} fts_2|_P \,)$. By Definition 4.11 this means $fts_1|_P \uplus fts_2|_P \models s_{*1} \leftrightarrows_{\mathrm{b}} s_{*2}$, and hence $(fts_1 \uplus fts_2)|_P \models s_{*1} \leftrightarrows_{\mathrm{b}} s_{*2}$ by Lemma 3.2, for all $P \in \mathcal{P}$ such that $P \models \varphi$.

By Theorem 4.14 this implies that $fts_1 \uplus fts_2 \models s_{*1} \overset{\varphi}{\leftrightarrows}_{bf} s_{*2}$.

By Definition 4.13 this is equivalent to $fts_1 \overset{\varphi}{\leftrightarrows}_{bf} fts_2$. □

Theorem 4.15. *Branching feature bisimilarity ($\overset{\varphi}{\leftrightarrows}_{bf}$) for given $\varphi \in \mathbb{B}(\mathcal{F})$ is an equivalence relation.*

Proof. We have to show that branching feature bisimilarity is reflexive, symmetric and transitive, for some feature expression $\varphi \in \mathbb{B}(\mathcal{F})$. Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS such that $s, t, u \in S$.

1. We prove reflexivity using s. By reflexivity of branching bisimilarity for LTS (Lemma 4.7), we have $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts \models s \leftrightarrows_{\mathrm{b}} s$. By Theorem 4.14 it follows that $fts \models s \overset{\varphi}{\leftrightarrows}_{bf} s$. Hence $\overset{\varphi}{\leftrightarrows}_{bf}$ is reflexive.

2. Suppose that $fts \models s \overset{\varphi}{\leftrightarrows}_{bf} t$. By Theorem 4.13 it follows that $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts|_P \models s \leftrightarrows_{\mathrm{b}} t$. By symmetry of branching bisimilarity for LTS (Lemma 4.7), we have $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts|_P \models t \leftrightarrows_{\mathrm{b}} s$. By Theorem 4.14 it follows that $fts \models t \overset{\varphi}{\leftrightarrows}_{bf} s$. Hence $\overset{\varphi}{\leftrightarrows}_{bf}$ is symmetric.

3. Suppose that $fts \models s \overset{\varphi}{\leftrightarrows}_{bf} t$ and $fts \models t \overset{\varphi}{\leftrightarrows}_{bf} u$. By Theorem 4.13 it follows that $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts|_P \models s \leftrightarrows_{\mathrm{b}} t \wedge fts|_P \models t \leftrightarrows_{\mathrm{b}} u$. By transitivity of branching bisimilarity for LTS (Lemma 4.7), we have $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts|_P \models s \leftrightarrows_{\mathrm{b}} u$. By Theorem 4.14 it follows that $fts \models s \overset{\varphi}{\leftrightarrows}_{bf} u$. Hence $\overset{\varphi}{\leftrightarrows}_{bf}$ is transitive.

□

Lemma 4.8. *Let fts be an FTS with states s, t and u such that $fts \models s \overset{\varphi}{\leftrightarrows}_{bf} t$ and $fts \models t \overset{\psi}{\leftrightarrows}_{bf} u$, for some $\varphi, \psi \in \mathbb{B}(\mathcal{F})$. Then $fts \models s \overset{\varphi\wedge\psi}{\longleftrightarrow}_{bf} u$.*

Proof. Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS such that $s, t, u \in S$, and such that $fts \models s \overset{\varphi}{\leftrightarrows}_{bf} t$ and $fts \models t \overset{\psi}{\leftrightarrows}_{bf} u$. By Theorem 4.13 it follows that $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fts|_P \models s \leftrightarrows_{\mathrm{b}} t$ and $\forall P \in \mathcal{P}\colon P \models \psi \Rightarrow fts|_P \models t \leftrightarrows_{\mathrm{b}} u$. By transitivity of branching bisimilarity for LTS (Lemma 4.7), we have $\forall P \in \mathcal{P}\colon P \models \varphi \wedge \psi \Rightarrow fts|_P \models s \leftrightarrows_{\mathrm{b}} u$. By Theorem 4.14 it follows that $fts \models s \overset{\varphi\wedge\psi}{\longleftrightarrow}_{bf} u$. □

Theorem 4.16. *A property is preserved by branching bisimulation for LTS if and only if this property is preserved by branching feature bisimulation for FTS.*

Proof. Let fts_1 and fts_2 be FTS such that $fts_1 \leftrightarrows_{bf} fts_2$, and let ϕ be a property that is preserved by branching bisimulation for LTS. We have to show that $fts_1 \models \phi \Leftrightarrow fts_2 \models \phi$.

Assume $fts_1 \models \phi$. By Definition 3.16 this is equivalent to $\forall P \in \mathcal{P}\colon fts_1|_P \models \phi$. Since $fts_1 \leftrightarrows_{bf} fts_2$, by Corollary 4.7 we have $\forall P \in \mathcal{P}\colon fts_1|_P \approx_{dbs} fts_2|_P$. Since ϕ is preserved by $\leftrightarrows_{\mathrm{b}}$, it follows that $\forall P \in \mathcal{P}\colon fts_2|_P \models \phi$, which is equivalent to $fts_2 \models \phi$, by Definition 3.16.

The proof for the implication in the other direction is similar.

□

Appendix C

Full proofs for Chapter 5

This appendix contains the full proofs that were omitted in Chapter 5.

Theorem 5.2. *For each FKS fks, and for all $\phi \in \mathbb{B}(\mathcal{F})$, it holds that $\overset{\phi}{\approx}_{dbsf}\text{-min}_{\texttt{FKS}}(fks) \overset{\phi}{\approx}_{dbsf} fks$.*

Proof. Let $fks = (S, AP, \theta, L, s_*)$ be an FKS, let $\phi \in \mathbb{B}(\mathcal{F})$ be a feature expression, and let $\overset{\phi}{\approx}_{dbsf}\text{-min}_{\texttt{FKS}}(fks) = (S', AP, \theta', L', s'_*)$ be the naive divergence-blind stuttering feature quotient for ϕ of fks. We define the relation R such that:

$$\begin{aligned} R = &\{(s, [\varphi \wedge \phi]_{\sim_{\mathcal{P}}}, C) \in S \times \widehat{\mathbb{B}}(\mathcal{F}) \times S' \mid \exists t \in C \colon fks \models s \overset{\varphi}{\approx}_{dbsf} t\} \cup \\ &\{(C, [\varphi \wedge \phi]_{\sim_{\mathcal{P}}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S \mid \exists t \in C \colon fks \models s \overset{\varphi}{\approx}_{dbsf} t\} \end{aligned}$$

We have to show that R is a divergence-blind stuttering feature bisimulation relation on $fks \uplus \overset{\phi}{\approx}_{dbsf}\text{-min}_{\texttt{FKS}}(fks)$ such that $(s_*, \hat{\phi}, s'_*) \in R$.

Since $\overset{\phi}{\approx}_{dbsf}$ is reflexive, we find that, for all $s \in S$, $(s, \widehat{\phi}, C) \in R$, where $C \in S'$ is such that $s \in C$. Using that $s_* \in s'_*$ by Definition 5.2, it immediately follows that $(s_*, \hat{\phi}, s'_*) \in R$. Hence, it remains to show that R is a divergence-blind stuttering feature bisimulation relation.

From the definition of R it immediately follows that it is symmetric, and hence it remains to show that R satisfies the conditions from Definition 4.9.

Consider a tuple $(s, [\varphi \wedge \phi]_{\sim_{\mathcal{P}}}, C) \in S \times \mathbb{B}(\mathcal{F}) \times S'$ such that $fks \models s \overset{\varphi}{\approx}_{dbsf} t$ and $t \in C$, for some $t \in S$. Let t be such.

We first show that $(L \uplus L')(s) = (L \uplus L')(C)$. Since $fks \models s \overset{\varphi}{\approx}_{dbsf} t$ we know that $L(s) = L(t)$, and since $t \in C$ we know that $L(t) = L'(C)$. Hence we can derive $(L \uplus L')(s) = L(s) = L(t) = L'(C) = (L \uplus L')(C)$.

Now we prove the transfer condition. Suppose that $s \xrightarrow{\psi} s'$, for some $\psi \in \mathbb{B}(\mathcal{F})$, and for some $s' \in S$. We first show that this transition can be mimicked by state t for all products satisfying $\varphi \wedge \psi$. Since $\mathit{fks} \models s \overset{\varphi}{\approx}_{dbsf} t$, there exists a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ such that $\mathit{fks} \models s \overset{\varphi'}{\approx}_{dbsf} t'$ and $\mathit{fks} \models s' \overset{\varphi''}{\approx}_{dbsf} t''$, for all $(t', \varphi', t'', \varphi'') \in \mathcal{T}$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t',\varphi',t'',\varphi'') \in \mathcal{T}} \check{\theta}_\tau(t,t') \wedge \varphi' \wedge (t' = t'' \vee \theta(t',t'')) \wedge \varphi''.$$

Now we show that C can mimic all transitions from t, which we use to conclude that C can mimic the transition from s to s'. Using that $t \in C$, from Definition 5.2 it follows that $\check{\theta}_\tau(t,t') \Rightarrow_{\mathcal{P}} \check{\theta}'_\tau(C, C_{t'})$ and $\theta(t',t'') \Rightarrow_{\mathcal{P}} \theta'(C_{t'}, C_{t''})$, with $t' \in C_{t'}$ and $t'' \in C_{t''}$, for all $(t', \varphi', t'', \varphi'') \in \mathcal{T}$. Since $t' \in C_{t'}$ and $\mathit{fks} \models s \overset{\varphi'}{\approx}_{dbsf} t'$, and $t'' \in C_{t''}$ and $\mathit{fks} \models s' \overset{\varphi''}{\approx}_{dbsf} t''$, by construction of R we have $(s, [\varphi' \wedge \phi]_{\sim_{\mathcal{P}}}, C_{t'}) \in R$ and $(s', [\varphi'' \wedge \phi]_{\sim_{\mathcal{P}}}, C_{t''}) \in R$, for all $(t', \varphi', t'', \varphi'') \in \mathcal{T}$. We find that

$$\begin{aligned} \phi \wedge \textstyle\bigvee_{(t',\varphi',t'',\varphi'') \in \mathcal{T}} \check{\theta}_\tau(t,t') \wedge \varphi' \wedge (t' = t'' \vee \theta(t',t'')) \wedge \varphi'' \;\Rightarrow_{\mathcal{P}} \\ \textstyle\bigvee_{(t',\varphi',t'',\varphi'') \in \mathcal{T}} \check{\theta}'_\tau(C, C_{t'}) \wedge \varphi' \wedge (C_{t'} = C_{t''} \vee \theta'(C_{t'}, C_{t''})) \wedge \varphi'' \wedge \phi, \end{aligned}$$

and hence the transfer condition for this tuple is satisfied.

We prove the transfer conditions for the remaining tuples of R in two steps, using a case distinction.

1. First consider a tuple $(C, \hat{\phi}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \in C$. From Definition 5.2 it follows immediately that $(L \uplus L')(C) = (L \uplus L')(s)$.

   Suppose that $C \xrightarrow{\psi} C'$, for some $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. From Definition 5.2 it follows that $\psi = \bigvee\{\theta(u,u') \mid u \in C \wedge u' \in C'\}$. Pick a pair $(u,u') \in C \times C'$. Since both $s \in C$ and $u \in C$, by Definition 5.2 we know that $\mathit{fks} \models s \overset{\phi}{\approx}_{dbsf} u$. Hence we can find a set $\mathcal{S}_{(u,u')} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ for each tuple $(u,u') \in C \times C'$ such that $\mathit{fks} \models s' \overset{\varphi'}{\approx}_{dbsf} u$ and $\mathit{fks} \models s'' \overset{\varphi''}{\approx}_{dbsf} u'$, for all $(s', \varphi', s'', \varphi'') \in \mathcal{S}_{(u,u')}$, and such that

   $$\phi \wedge \theta(u,u') \;\Rightarrow_{\mathcal{P}} \bigvee_{(s',\varphi',s'',\varphi'') \in \mathcal{S}_{(u,u')}} \check{\theta}_\tau(s,s') \wedge \varphi' \wedge (s' = s'' \vee \theta(s',s'')) \wedge \varphi''.$$

   For each tuple $(u,u') \in C \times C'$ we have that $\mathit{fks} \models s' \overset{\varphi'}{\approx}_{dbsf} u$ and $\mathit{fks} \models s'' \overset{\varphi''}{\approx}_{dbsf} u'$, and hence by construction of R we have $(C, [\varphi' \wedge \phi]_{\sim_{\mathcal{P}}}, s') \in R$ and $(C', [\varphi'' \wedge \phi]_{\sim_{\mathcal{P}}}, s'') \in R$, for all $(s', \varphi', s'', \varphi'') \in \mathcal{S}_{(u,u')}$. We find that

   $$\phi \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(u,u') \in C \times C'} \bigvee_{(s',\varphi',s'',\varphi'') \in \mathcal{T}_{(u,u')}} \check{\theta}_\tau(s,s') \wedge \varphi' \wedge (s' = s'' \vee \theta(s',s'')) \wedge \varphi'' \wedge \phi,$$

   and hence the transfer condition for this tuple is satisfied.

2. Lastly consider a tuple $(C, [\varphi \wedge \phi]_{\sim_{\mathcal{P}}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \notin C$, and such that $\mathit{fks} \models s \overset{\varphi}{\approx}_{\mathit{dbsf}} t$ and $t \in C$, for some $t \in S$. Let t be such.

   We first show that $(L \uplus L')(C) = (L \uplus L')(s)$. Since $\mathit{fks} \models s \overset{\varphi}{\approx}_{\mathit{dbsf}} t$ we know that $L(s) = L(t)$, and since $t \in C$ we know that $L(t) = L'(C)$. Hence we can derive $(L \uplus L')(C) = L'(C) = L(t) = L(s) = (L \uplus L')(s)$.

   Suppose that $C \xrightarrow{\psi} C'$, for some $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. By construction of R we have $(C, \hat{\phi}, t) \in R$. We have already shown that the transfer conditions are satisfied for such pairs in R, and hence we can find a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ such that $(C, [\phi \wedge \varphi'_t]_{\sim_{\mathcal{P}}}, t') \in R$ and $(C', [\phi \wedge \varphi''_t]_{\sim_{\mathcal{P}}}, t'') \in R$, for all $(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}$, and such that

$$\phi \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}} \check{\theta}_\tau(t, t') \wedge \varphi'_t \wedge (t' = t'' \vee \theta(t', t'')) \wedge \varphi''_t \wedge \phi.$$

   Now we show that s can mimic all transitions from t, which we use to conclude that s can mimic the transition from C to C'. Since $\mathit{fks} \models s \overset{\varphi}{\approx}_{\mathit{dbsf}} t$, we can find sets $\mathcal{S}_{(t', t'')} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ for each $(t', \varphi'_t, t'', \varphi'') \in \mathcal{T}$ such that, for all $(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{(t', t'')}$, we have $\mathit{fks} \models s' \overset{\varphi'_s}{\approx}_{\mathit{dbsf}} t'$ and $\mathit{fks} \models s'' \overset{\varphi''_s}{\approx}_{\mathit{dbsf}} t''$, and such that

$$\varphi \wedge \check{\theta}_\tau(t, t') \wedge (t' = t'' \vee \theta(t', t'')) \;\Rightarrow_{\mathcal{P}} \bigvee_{(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{t', t''}} \check{\theta}_\tau(s, s') \wedge \varphi'_s \wedge (s' = s'' \vee \theta(s', s'')) \wedge \varphi''_s.$$

   Since $(C, [\phi \wedge \varphi'_t]_{\sim_{\mathcal{P}}}, t') \in R$ and $(C, [\phi \wedge \varphi''_t]_{\sim_{\mathcal{P}}}, t'') \in R$, for all $(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}$, by definition of R this means that $\mathit{fks} \models t' \overset{\varphi'_t}{\approx}_{\mathit{dbsf}} u$ and $\mathit{fks} \models t'' \overset{\varphi''_t}{\approx}_{\mathit{dbsf}} u'$, for some $u \in C$, $u', \in C'$. Since $\mathit{fks} \models s' \overset{\varphi'_s}{\approx}_{\mathit{dbsf}} t'$ and $\mathit{fks} \models s'' \overset{\varphi''_s}{\approx}_{\mathit{dbsf}} t''$ for all $(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{(t', t'')}$, we derive $\mathit{fks} \models s' \overset{\varphi'_s \wedge \varphi'_t}{\approx}_{\mathit{dbsf}} u$ and $\mathit{fks} \models s'' \overset{\varphi''_s \wedge \varphi''_t}{\approx}_{\mathit{dbsf}} u'$ using productwise transitivity. By construction of R we have $(C, [\phi \wedge \varphi'_s \wedge \varphi'_t]_{\sim_{\mathcal{P}}}, s') \in R$ and $(C', [\phi \wedge \varphi''_s \wedge \varphi''_t]_{\sim_{\mathcal{P}}}, s'') \in R$. We find that

$$\varphi \wedge \textstyle\bigvee_{(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}} \check{\theta}_\tau(t, t') \wedge \varphi'_t \wedge (t' = t'' \vee \theta(t', t'')) \wedge \varphi''_t \wedge \phi \;\Rightarrow_{\mathcal{P}}$$
$$\textstyle\bigvee_{(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}} \bigvee_{(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{t', t''}} \check{\theta}_\tau(s, s') \wedge \varphi'_s \wedge \varphi'_t \wedge (s' = s'' \vee \theta(s', s'')) \wedge \varphi''_s \wedge \varphi''_t \wedge \phi,$$

   and hence that

$$\varphi \wedge \phi \wedge \psi \;\Rightarrow_{\mathcal{P}}$$
$$\textstyle\bigvee_{(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}} \bigvee_{(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{t', t''}} \check{\theta}_\tau(s, s') \wedge \varphi'_s \wedge \varphi'_t \wedge (s' = s'' \vee \theta(s', s'')) \wedge \varphi''_s \wedge \varphi''_t \wedge \phi,$$

   from which we conclude that the transfer condition for this pair is also satisfied.

□

Theorem 5.3. *For each FKS fks and for all* $\mathit{fks}' \in \leftrightarrows_{cf}\text{-min}_{\mathtt{FKS}}(\mathit{fks})$*, it holds that* $\mathit{fks}' \leftrightarrows_f \mathit{fks}$*.*

Proof. Let $\mathit{fks} = (S, AP, \theta, L, s_*)$ be an FKS, let $\phi \in \mathbb{B}(\mathcal{F})$ be a feature expression, and let $\mathit{fks}' = (S', AP, \theta', L', s'_*)$ be a coherent feature bisimulation quotient of fks. We define the relation R such that:

$$R = \{(s, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_{\mathcal{P}}}, C) \in S \times \widehat{\mathbb{B}}(\mathcal{F}) \times S' \mid \exists t \in C \colon \mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t\} \cup$$
$$\{(C, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_{\mathcal{P}}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S \mid \exists t \in C \colon \mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t\}$$

We have to show that R is a feature bisimulation relation on $\mathit{fks} \uplus \mathit{fks}'$ such that $(s_*, \widehat{\mathsf{true}}, s'_*) \in R$.

Since $\leftrightarrows_f$ is reflexive, we find that, for all $s \in S$, $(s, \widehat{\varrho(s)}, C) \in R$, where $C \in S'$ is such that $s \in C$. Using that $s_* \in s'_*$ by Definition 5.4, it immediately follows that $(s_*, \mathsf{true}, s'_*) \in R$. Hence, it remains to show that R is a feature bisimulation relation.

From the definition of R it immediately follows that it is symmetric, and hence it remains to show that R satisfies the conditions from Definition 4.2.

Consider a tuple $(s, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_{\mathcal{P}}}, C) \in S \times \mathbb{B}(\mathcal{F}) \times S'$ such that $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$ and $t \in C$, for some $t \in S$. Let t be such.

We first show that $(L \uplus L')(s) = (L \uplus L')(C)$. Since $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$ we know that $L(s) = L(t)$, and since $t \in C$ we know that $L(t) = L'(C)$. Hence we can derive $(L \uplus L')(s) = L(s) = L(t) = L'(C) = (L \uplus L')(C)$.

Now we prove the transfer condition. Suppose that $s \xrightarrow{\psi} s'$, for some $\psi \in \mathbb{B}(\mathcal{F})$, and for some $s' \in S$. We first show that this transition can be mimicked by state t for all products satisfying $\varphi \wedge \psi$. Since $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$, there exists a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F})$ such that $\mathit{fks} \models s' \overset{\varphi'}{\leftrightarrows}_f t'$, for all $(t', \varphi') \in \mathcal{T}$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t', \varphi') \in \mathcal{T}} \theta(t, t') \wedge \varphi'.$$

Now we show that C can mimic all transitions from t, which we use to conclude that C can mimic the transition from s to s'. Using that $t \in C$, from Definition 5.4 it follows that $(\theta(t, t') \wedge \varrho(t)) \Rightarrow_{\mathcal{P}} \theta'(C, C_{t'})$, with $t' \in C_{t'}$, for all $(t', \varphi') \in \mathcal{T}$. Since $t' \in C_{t'}$ and $\mathit{fks} \models s' \overset{\varphi'}{\leftrightarrows}_f t'$, by construction of R we have $(s', [\varphi' \wedge \varrho(s') \wedge \varrho(t')]_{\sim_{\mathcal{P}}}, C_{t'}) \in R$, for all $(t', \varphi') \in \mathcal{T}$. Using that $\varrho(s) \wedge \psi \Rightarrow_{\mathcal{P}} \varrho(s')$ and $\varrho(t) \wedge \theta(t, t') \Rightarrow_{\mathcal{P}} \varrho(t')$, for all $(t', \varphi') \in \mathcal{T}$, by definition of reachability, we find that

$$\varrho(s) \wedge \varrho(t) \wedge \bigvee_{(t', \varphi') \in \mathcal{T}} \theta(t, t') \wedge \varphi' \;\Rightarrow_{\mathcal{P}} \bigvee_{(t', \varphi') \in \mathcal{T}} \theta'(C, C_{t'}) \wedge \varphi' \wedge \varrho(s') \wedge \varrho(t'),$$

and hence the transfer condition for this tuple is satisfied.

We prove the transfer conditions for the remaining tuples of R in two steps, using a case distinction.

1. First consider a tuple $(C, \widehat{\varrho(s)}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \in C$. From Definition 5.4 it follows immediately that $(L \uplus L')(C) = (L \uplus L')(s)$.

Suppose that $C \xrightarrow{\psi} C'$, for some $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. From Definition 5.4 it follows that $\psi = \bigvee\{\theta(u,u') \wedge \varrho(u) \mid u \in C \wedge u' \in C'\}$. Pick a pair $(u,u') \in C \times C'$. Since both $s \in C$ and $u \in C$, by Definition 5.4 we know that $\mathit{fks} \models s \leftrightarrows_{cf} u$. Hence we can find a set $\mathcal{S}_{(u,u')} \subseteq S \times \mathbb{B}(\mathcal{F})$ for each tuple $(u,u') \in C \times C'$ such that $\mathit{fks} \models s' \overset{\varphi'}{\leftrightarrows}_f u'$, for all $(s',\varphi') \in \mathcal{S}_{(u,u')}$, and such that

$$\varrho(s) \wedge \varrho(u) \wedge \theta(u,u') \;\Rightarrow_{\mathcal{P}} \bigvee_{(s',\varphi') \in \mathcal{S}_{(u,u')}} \theta(s,s') \wedge \varphi'.$$

For each tuple $(u,u') \in C \times C'$ we have that $\mathit{fks} \models s' \overset{\varphi'}{\leftrightarrows}_f u'$, and hence by construction of R we have $(C', [\varphi' \wedge \varrho(s') \wedge \varrho(u')]_{\sim_{\mathcal{P}}}, s') \in R$, for all $(s',\varphi') \in \mathcal{S}_{(u,u')}$. Using that $\varrho(s) \wedge \theta(s,s') \Rightarrow_{\mathcal{P}} \varrho(s')$ and $\varrho(u) \wedge \theta(u,u') \Rightarrow_{\mathcal{P}} \varrho(u')$, we find that

$$\varrho(s) \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(u,u') \in C \times C'} \bigvee_{(s',\varphi') \in \mathcal{T}_{(u,u')}} \theta(s,s') \wedge \varphi' \wedge \varrho(s') \wedge \varrho(u'),$$

and hence the transfer condition for this tuple is satisfied.

2. Lastly consider a tuple $(C, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_{\mathcal{P}}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \notin C$, and such that $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$ and $t \in C$, for some $t \in S$. Let t be such.

   We first show that $(L \uplus L')(C) = (L \uplus L')(s)$. Since $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$ we know that $L(s) = L(t)$, and since $t \in C$ we know that $L(t) = L'(C)$. Hence we can derive $(L \uplus L')(C) = L'(C) = L(t) = L(s) = (L \uplus L')(s)$.

   Suppose that $C \xrightarrow{\psi} C'$, for some $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. By construction of R we have $(C, \widehat{\varrho(t)}, t) \in R$. We have already shown that the transfer conditions are satisfied for such pairs in R, and hence we can find a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F})$ such that $(C', \hat{\varphi}'_t, t') \in R$, for all $(t', \varphi'_t) \in \mathcal{T}$, and such that

$$\varrho(t) \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t',\varphi'_t) \in \mathcal{T}} \theta(t,t') \wedge \varphi'_t.$$

   Now we show that s can mimic all transitions from t, which we use to conclude that s can mimic the transition from C to C'. Since $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$, we can find sets $\mathcal{S}_{t'} \subseteq S \times \mathbb{B}(\mathcal{F})$ for each $(t', \varphi'_t) \in \mathcal{T}$ such that, for all $(s', \varphi'_s) \in \mathcal{S}_{t'}$, we have $\mathit{fks} \models s' \overset{\varphi'_s}{\leftrightarrows}_f t'$ and such that

$$\varphi \wedge \theta(t,t') \;\Rightarrow_{\mathcal{P}} \bigvee_{(s',\varphi'_s) \in \mathcal{S}_{t'}} \theta(s,s') \wedge \varphi'_s.$$

Since $(C', \hat{\varphi}'_t, t') \in R$, for all $(t', \varphi'_t) \in \mathcal{T}$, by definition of R this means that $\varphi'_t \sim_{\mathcal{P}} \varphi''_t \wedge \varrho(t') \wedge \varrho(u_{t'})$, for some $\varphi''_t \in \mathbb{B}(\mathcal{F})$ and some $u_{t'} \in C'$, such that $fks \models t' \xleftrightarrow{\varphi''_t}_f u_{t'}$. Since $fks \models s' \xleftrightarrow{\varphi'_s}_f t'$ for all $(s', \varphi'_s) \in \mathcal{S}_{t'}$, we derive $fks \models s' \xleftrightarrow{\varphi'_s \wedge \varphi''_t}_f u_{t'}$ using productwise transitivity. By construction of R we have $(C', [\varphi'_s \wedge \varphi''_t \wedge \varrho(s') \wedge \varrho(u_{t'})]_{\sim_{\mathcal{P}}}, s') \in R$. Using that $\varrho(s) \wedge \psi \Rightarrow_{\mathcal{P}} \varrho(s')$ and $\varrho'_t \Rightarrow_{\mathcal{P}} \varrho''_t \wedge \varrho(u_{t'})$, we find that We find that

$$\varphi \wedge \bigvee_{(t', \varphi'_t) \in \mathcal{T}} \theta(t, t') \wedge \varphi'_t \;\Rightarrow_{\mathcal{P}} \bigvee_{(t', \varphi'_t) \in \mathcal{T}} \bigvee_{(s', \varphi'_s) \in \mathcal{S}_{t'}} \theta(s, s') \wedge \varphi'_s \wedge \varphi'_t,$$

and hence that

$$\varphi \wedge \varrho(s) \wedge \varrho(t) \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t', \varphi'_t) \in \mathcal{T}} \bigvee_{(s', \varphi'_s) \in \mathcal{S}_{t'}} \theta(s, s') \wedge \varphi'_s \wedge \varphi''_t \wedge \varrho(s') \wedge \varrho(u_{t'}),$$

from which we conclude that the transfer condition for this pair is also satisfied.

□

Theorem 5.4. *For each FKS fks and for all $fks' \in \approx_{cdbsf}\text{-min}_{\mathtt{FKS}}(fks)$, it holds that $fks' \approx_{dbsf} fks$.*

Proof. Let $fks = (S, AP, \theta, L, s_*)$ be an FKS, let $\phi \in \mathbb{B}(\mathcal{F})$ be a feature expression, and let $fks' = (S', AP, \theta', L', s'_*)$ be a coherent divergence-blind stuttering feature quotient of fks. We define the relation R such that:

$$\begin{aligned} R = &\{(s, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_{\mathcal{P}}}, C) \in S \times \widehat{\mathbb{B}}(\mathcal{F}) \times S' \mid \exists t \in C \colon fks \models s \overset{\varphi}{\approx}_{dbsf} t\} \cup \\ &\{(C, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_{\mathcal{P}}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S \mid \exists t \in C \colon fks \models s \overset{\varphi}{\approx}_{dbsf} t\} \end{aligned}$$

We have to show that R is a divergence-blind stuttering feature bisimulation relation on $fks \uplus fks'$ such that $(s_*, \widehat{\mathsf{true}}, s'_*) \in R$.

Since $\approx_{dbsf}$ is reflexive, we find that, for all $s \in S$, $(s, \widehat{\varrho(s)}, C) \in R$, where $C \in S'$ is such that $s \in C$. Using that $s_* \in s'_*$ by Definition 5.4, it immediately follows that $(s_*, \widehat{\mathsf{true}}, s'_*) \in R$. Hence, it remains to show that R is a divergence-blind stuttering feature bisimulation relation.

From the definition of R it immediately follows that it is symmetric, and hence it remains to show that R satisfies the conditions from Definition 4.9.

Consider a tuple $(s, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_{\mathcal{P}}}, C) \in S \times \mathbb{B}(\mathcal{F}) \times S'$ such that $fks \models s \overset{\varphi}{\approx}_{dbsf} t$ and $t \in C$, for some $t \in S$. Let t be such.

We first show that $(L \uplus L')(s) = (L \uplus L')(C)$. Since $fks \models s \overset{\varphi}{\approx}_{dbsf} t$ we know that $L(s) = L(t)$, and since $t \in C$ we know that $L(t) = L'(C)$. Hence we can derive $(L \uplus L')(s) = L(s) = L(t) = L'(C) = (L \uplus L')(C)$.

Now we prove the transfer condition. Suppose that $s \xrightarrow{\psi} s'$, for some $\psi \in \mathbb{B}(\mathcal{F})$, and for some $s' \in S$. We first show that this transition can be mimicked by state t for all products satisfying $\varphi \wedge \psi$. Since $fks \models s \overset{\varphi}{\approx}_{dbsf} t$, there exists a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ such that $fks \models s \overset{\varphi'}{\approx}_{dbsf} t'$ and $fks \models s' \overset{\varphi''}{\approx}_{dbsf} t''$, for all $(t', \varphi', t'', \varphi'') \in \mathcal{T}$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t', \varphi', t'', \varphi'') \in \mathcal{T}} \check{\theta}_\tau(t, t') \wedge \varphi' \wedge (t' = t'' \vee \theta(t', t'')) \wedge \varphi''.$$

Now we show that C can mimic all transitions from t, which we use to conclude that C can mimic the transition from s to s'. Using that $t \in C$, from Definition 5.4 it follows that $(\check{\theta}_\tau(t,t') \wedge \varrho(t)) \Rightarrow_{\mathcal{P}} \check{\theta}'_\tau(C, C_{t'})$ and $(\theta(t',t'') \wedge \varrho(t')) \Rightarrow_{\mathcal{P}} \theta'(C_{t'}, C_{t''})$, with $t' \in C_{t'}$ and $t'' \in C_{t''}$, for all $(t', \varphi', t'', \varphi'') \in \mathcal{T}$. Since $t' \in C_{t'}$ and $\mathit{fks} \models s \overset{\varphi'}{\approx}_{dbsf} t'$, and $t'' \in C_{t''}$ and $\mathit{fks} \models s' \overset{\varphi''}{\approx}_{dbsf} t''$, by construction of R we have $(s, [\varphi' \wedge \varrho(s) \wedge \varrho(t')]_{\sim_{\mathcal{P}}}, C_{t'}) \in R$ and $(s', [\varphi'' \wedge \varrho(s') \wedge \varrho(t'')]_{\sim_{\mathcal{P}}}, C_{t''}) \in R$, for all $(t', \varphi', t'', \varphi'') \in \mathcal{T}$. Using that $\varrho(s) \wedge \psi \Rightarrow_{\mathcal{P}} \varrho(s')$ and $\varrho(t) \wedge \check{\theta}_\tau(t,t') \wedge (t' = t'' \vee \theta(t',t'')) \Rightarrow_{\mathcal{P}} \varrho(t) \wedge \varrho(t')$, for all $(t', \varphi', t'', \varphi'') \in \mathcal{T}$, by definition of reachability, we find that

$$\varrho(s) \wedge \varrho(t) \wedge \bigvee_{(t',\varphi',t'',\varphi'') \in \mathcal{T}} \check{\theta}_\tau(t,t') \wedge \varphi' \wedge (t' = t'' \vee \theta(t',t'')) \wedge \varphi'' \Rightarrow_{\mathcal{P}} \bigvee_{(t',\varphi',t'',\varphi'') \in \mathcal{T}} \check{\theta}'_\tau(C, C_{t'}) \wedge \varphi' \wedge \varrho(s) \wedge \varrho(t') \wedge (C_{t'} = C_{t''} \vee \theta'(C_{t'}, C_{t''})) \wedge \varphi'' \wedge \varrho(s') \wedge \varrho(t''),$$

and hence the transfer condition for this tuple is satisfied.

We prove the transfer conditions for the remaining tuples of R in two steps, using a case distinction.

1. First consider a tuple $(C, \widehat{\varrho(s)}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \in C$. From Definition 5.4 it follows immediately that $(L \uplus L')(C) = (L \uplus L')(s)$.

   Suppose that $C \xrightarrow{\psi} C'$, for some $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. From Definition 5.4 it follows that $\psi = \bigvee\{\theta(u,u') \wedge \varrho(u) \mid u \in C \wedge u' \in C'\}$. Pick a pair $(u,u') \in C \times C'$. Since both $s \in C$ and $u \in C$, by Definition 5.4 we know that $\mathit{fks} \models s \approx_{cdbsf} u$. Hence we can find a set $\mathcal{S}_{(u,u')} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ for each tuple $(u,u') \in C \times C'$ such that $\mathit{fks} \models s' \overset{\varphi'}{\approx}_{dbsf} u$ and $\mathit{fks} \models s'' \overset{\varphi''}{\approx}_{dbsf} u'$, for all $(s', \varphi', s'', \varphi'') \in \mathcal{S}_{(u,u')}$, and such that

   $$\varrho(s) \wedge \varrho(u) \wedge \theta(u,u') \Rightarrow_{\mathcal{P}} \bigvee_{(s',\varphi',s'',\varphi'') \in \mathcal{S}_{(u,u')}} \check{\theta}_\tau(s,s') \wedge \varphi' \wedge (s' = s'' \vee \theta(s',s'')) \wedge \varphi''.$$

   For each tuple $(u,u') \in C \times C'$ we have that $\mathit{fks} \models s' \overset{\varphi'}{\approx}_{dbsf} u$ and $\mathit{fks} \models s'' \overset{\varphi''}{\approx}_{dbsf} u'$, and hence by construction of R we have $(C, [\varphi' \wedge \varrho(s') \wedge \varrho(u)]_{\sim_{\mathcal{P}}}, s') \in R$ and $(C', [\varphi'' \wedge \varrho(s'') \wedge \varrho(u')]_{\sim_{\mathcal{P}}}, s'') \in R$, for all $(s', \varphi', s'', \varphi'') \in \mathcal{S}_{(u,u')}$. Using that $\varrho(s) \wedge \check{\theta}_\tau(s,s') \wedge (s' = s'' \vee \theta(s',s'')) \Rightarrow_{\mathcal{P}} \varrho(s') \wedge \varrho(s'')$ and $\varrho(u) \wedge \theta(u,u') \Rightarrow_{\mathcal{P}} \varrho(u')$, we find that

   $$\varrho(s) \wedge \psi \Rightarrow_{\mathcal{P}} \bigvee_{(u,u') \in C \times C'} \bigvee_{(s',\varphi',s'',\varphi'') \in \mathcal{T}_{(u,u')}} \check{\theta}_\tau(s,s') \wedge \varphi' \wedge \varrho(s') \wedge \varrho(u) \wedge (s' = s'' \vee \theta(s',s'')) \wedge \varphi'' \wedge \varrho(s'') \wedge \varrho(u'),$$

   and hence the transfer condition for this tuple is satisfied.

2. Lastly consider a tuple $(C, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_{\mathcal{P}}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \notin C$, and such that $\mathit{fks} \models s \overset{\varphi}{\approx}_{dbsf} t$ and $t \in C$, for some $t \in S$. Let t be such.

   We first show that $(L \uplus L')(C) = (L \uplus L')(s)$. Since $\mathit{fks} \models s \overset{\varphi}{\approx}_{dbsf} t$ we know that $L(s) = L(t)$, and since $t \in C$ we know that $L(t) = L'(C)$. Hence we can derive $(L \uplus L')(C) = L'(C) = L(t) = L(s) = (L \uplus L')(s)$.

Suppose that $C \xrightarrow{\psi} C'$, for some $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. By construction of R we have $(C, \widehat{\varrho(t)}, t) \in R$. We have already shown that the transfer conditions are satisfied for such pairs in R, and hence we can find a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ such that $(C, \hat{\varphi}'_t, t') \in R$ and $(C', \hat{\varphi}''_t, t'') \in R$, for all $(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}$, and such that

$$\varrho(t) \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}} \check{\theta}_\tau(t, t') \wedge \varphi'_t \wedge (t' = t'' \vee \theta(t', t'')) \wedge \varphi''_t.$$

Now we show that s can mimic all transitions from t, which we use to conclude that s can mimic the transition from C to C'. Since $\mathit{fks} \models s \overset{\varphi}{\approx}_{dbsf} t$, we can find sets $\mathcal{S}_{(t', t'')} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ for each $(t', \varphi'_t, t'', \varphi'') \in \mathcal{T}$ such that, for all $(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{(t', t'')}$, we have $\mathit{fks} \models s' \overset{\varphi'_s}{\approx}_{dbsf} t'$ and $\mathit{fks} \models s'' \overset{\varphi''_s}{\approx}_{dbsf} t''$, and such that

$$\varphi \wedge \check{\theta}_\tau(t, t') \wedge (t' = t'' \vee \theta(t', t'')) \;\Rightarrow_{\mathcal{P}} \bigvee_{(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{t', t''}} \check{\theta}_\tau(s, s') \wedge \varphi'_s \wedge (s' = s'' \vee \theta(s', s'')) \wedge \varphi''_s.$$

Since $(C, \hat{\varphi}'_t, t') \in R$ and $(C', \hat{\varphi}''_t, t'') \in R$, for all $(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}$, by definition of R this means that $\varphi'_t \sim_{\mathcal{P}} \varphi^*_t \wedge \varrho(t') \wedge \varrho(u_{t'})$ and $\varphi''_t \sim_{\mathcal{P}} \varphi^{**}_t \wedge \varrho(t'') \wedge \varrho(u_{t''})$, for some $\varphi^*_t, \varphi^{**}_t \in \mathbb{B}(\mathcal{F})$ and some $u_{t'} \in C$, $u_{t''} \in C'$, such that $\mathit{fks} \models t' \overset{\varphi^*_t}{\approx}_{dbsf} u_{t'}$ and $\mathit{fks} \models t'' \overset{\varphi^{**}_t}{\approx}_{dbsf} u_{t''}$. Since $\mathit{fks} \models s' \overset{\varphi'_s}{\approx}_{dbsf} t'$ and $\mathit{fks} \models s'' \overset{\varphi''_s}{\approx}_{dbsf} t''$ for all $(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{(t', t'')}$, we derive $\mathit{fks} \models s' \overset{\varphi'_s \wedge \varphi^*_t}{\approx}_{dbsf} u_{t'}$ and $\mathit{fks} \models s'' \overset{\varphi''_s \wedge \varphi^{**}_t}{\approx}_{dbsf} u_{t''}$ using productwise transitivity. By construction of R we have $(C, [\varphi'_s \wedge \varphi^*_t \wedge \varrho(s') \wedge \varrho(u_{t'})]_{\sim_{\mathcal{P}}}, s') \in R$ and $(C', [\varphi''_s \wedge \varphi^{**}_t \wedge \varrho(s'') \wedge \varrho(u_{t''})]_{\sim_{\mathcal{P}}}, s'') \in R$. Using that $\varrho(s) \wedge \check{\theta}_\tau(s, s') \wedge (s' = s'' \vee \theta(s', s'')) \Rightarrow_{\mathcal{P}} \varrho(s') \wedge \varrho(s'')$ and $\varphi'_t \Rightarrow_{\mathcal{P}} \varphi^*_t \wedge \varrho(u_{t'})$ and $\varphi''_t \Rightarrow_{\mathcal{P}} \varphi^{**}_t \wedge \varrho(u_{t''})$, we find that

$$\begin{array}{l} \varphi \wedge \bigvee_{(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}} \check{\theta}_\tau(t, t') \wedge \varphi'_t \wedge (t' = t'' \vee \theta(t', t'')) \wedge \varphi''_t \;\Rightarrow_{\mathcal{P}} \\ \bigvee_{(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}} \bigvee_{(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{t', t''}} \check{\theta}_\tau(s, s') \wedge \varphi'_s \wedge \varphi'_t \wedge (s' = s'' \vee \theta(s', s'')) \wedge \varphi''_s \wedge \varphi''_t, \end{array}$$

and hence that

$$\begin{array}{l} \varphi \wedge \varrho(s) \wedge \varrho(t) \wedge \psi \;\Rightarrow_{\mathcal{P}} \\ \quad \bigvee_{(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}} \bigvee_{(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{t', t''}} \check{\theta}_\tau(s, s') \wedge \varphi'_s \wedge \varphi^*_t \wedge \varrho(s') \wedge \varrho(u_{t'}) \wedge \\ \qquad (s' = s'' \vee \theta(s', s'')) \wedge \varphi''_s \wedge \varphi^{**}_t \wedge \varrho(s'') \wedge \varrho(u_{t''}), \end{array}$$

from which we conclude that the transfer condition for this pair is also satisfied.

□

Theorem 5.5. *For each FTS fts, and for all $\phi \in \mathbb{B}(\mathcal{F})$, it holds that $\overset{\phi}{\leftrightarrows}_f$-$\min_{\mathtt{FTS}}(\mathit{fts}) \overset{\phi}{\leftrightarrows}_f \mathit{fts}$.*

Proof. Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, let $\phi \in \mathbb{B}(\mathcal{F})$ be a feature expression, and let $\overset{\phi}{\leftrightarrows}_f$ -$\min_{\mathtt{FTS}}(fts) = (S', \mathcal{A}, \theta', s'_*)$ be the naive feature bisimulation quotient for ϕ of fts. We define the relation R such that:

$$R = \{(s, [\varphi \wedge \phi]_{\sim_\mathcal{P}}, C) \in S \times \widehat{\mathbb{B}}(\mathcal{F}) \times S' \mid \exists t \in C \colon fts \models s \overset{\varphi}{\leftrightarrows}_f t\} \cup$$
$$\{(C, [\varphi \wedge \phi]_{\sim_\mathcal{P}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S \mid \exists t \in C \colon fts \models s \overset{\varphi}{\leftrightarrows}_f t\}$$

We have to show that R is a feature bisimulation relation on $fts \uplus \overset{\phi}{\leftrightarrows}_f$-$\min_{\mathtt{FTS}}(fts)$ such that $(s_*, \hat{\phi}, s'_*) \in R$.

Since $\overset{\phi}{\leftrightarrows}_f$ is reflexive, we find that, for all $s \in S$, $(s, \widehat{\phi}, C) \in R$, where $C \in S'$ is such that $s \in C$. Using that $s_* \in s'_*$ by Definition 5.6, it immediately follows that $(s_*, \hat{\phi}, s'_*) \in R$. Hence, it remains to show that R is a feature bisimulation relation.

From the definition of R it immediately follows that it is symmetric, and hence it remains to show that R satisfies the conditions from Definition 4.5.

Consider a tuple $(s, [\varphi \wedge \phi]_{\sim_\mathcal{P}}, C) \in S \times \mathbb{B}(\mathcal{F}) \times S'$ such that $fts \models s \overset{\varphi}{\leftrightarrows}_f t$ and $t \in C$, for some $t \in S$. Let t be such.

Suppose that $s \xrightarrow{\alpha|\psi} s'$, for some $\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$, and for some $s' \in S$. We first show that this transition can be mimicked by state t for all products satisfying $\varphi \wedge \psi$. Since $fts \models s \overset{\varphi}{\leftrightarrows}_f t$, there exists a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F})$ such that $fts \models s' \overset{\varphi'}{\leftrightarrows}_f t'$, for all $(t', \varphi') \in \mathcal{T}$, and such that

$$\varphi \wedge \psi \;\Rightarrow_\mathcal{P} \bigvee_{(t', \varphi') \in \mathcal{T}} \theta(t, \alpha, t') \wedge \varphi'.$$

Now we show that C can mimic all transitions from t, which we use to conclude that C can mimic the transition from s to s'. Using that $t \in C$, from Definition 5.6 it follows that $\theta(t, \alpha, t') \Rightarrow_\mathcal{P} \theta'(C, \alpha, C_{t'})$, with $t' \in C_{t'}$, for all $(t', \varphi') \in \mathcal{T}$. Since $t' \in C_{t'}$ and $fts \models s' \overset{\varphi'}{\leftrightarrows}_f t'$, by construction of R we have $(s', [\varphi' \wedge \phi]_{\sim_\mathcal{P}}, C_{t'}) \in R$, for all $(t', \varphi') \in \mathcal{T}$. We find that

$$\phi \wedge \bigvee_{(t', \varphi') \in \mathcal{T}} \theta(t, \alpha, t') \wedge \varphi' \;\Rightarrow_\mathcal{P} \bigvee_{(t', \varphi') \in \mathcal{T}} \theta'(C, \alpha, C_{t'}) \wedge \varphi' \wedge \phi,$$

and hence the transfer condition for this tuple is satisfied.

We prove the transfer conditions for the remaining tuples of R in two steps, using a case distinction.

1. First consider a tuple $(C, \hat{\phi}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \in C$.

Suppose that $C \xrightarrow{\alpha|\psi} C'$, for some $\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. From Definition 5.6 it follows that $\psi = \bigvee\{\theta(u,\alpha,u') \mid u \in C \wedge u' \in C'\}$. Pick a pair $(u,u') \in C \times C'$. Since both $s \in C$ and $u \in C$, by Definition 5.6 we know that $fts \models s \overset{\phi}{\leftrightarrows}_f u$. Hence we can find a set $\mathcal{S}_{(u,u')} \subseteq S \times \mathbb{B}(\mathcal{F})$ for each tuple $(u,u') \in C \times C'$ such that $fts \models s' \overset{\varphi'}{\leftrightarrows}_f u'$, for all $(s',\varphi') \in \mathcal{S}_{(u,u')}$, and such that

$$\phi \wedge \theta(u,\alpha,u') \;\Rightarrow_{\mathcal{P}} \bigvee_{(s',\varphi') \in \mathcal{S}_{(u,u')}} \theta(s,\alpha,s') \wedge \varphi'.$$

For each tuple $(u,u') \in C \times C'$ we have that $fts \models s' \overset{\varphi'}{\leftrightarrows}_f u'$, and hence by construction of R we have $(C', [\varphi' \wedge \phi]_{\sim_{\mathcal{P}}}, s') \in R$, for all $(s',\varphi') \in \mathcal{S}_{(u,u')}$. We find that

$$\phi \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(u,u') \in C \times C'} \bigvee_{(s',\varphi') \in \mathcal{T}_{(u,u')}} \theta(s,\alpha,s') \wedge \varphi' \wedge \phi,$$

and hence the transfer condition for this tuple is satisfied.

2. Lastly consider a tuple $(C, [\varphi \wedge \phi]_{\sim_{\mathcal{P}}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \notin C$, and such that $fts \models s \overset{\varphi}{\leftrightarrows}_f t$ and $t \in C$, for some $t \in S$. Let t be such.

   Suppose that $C \xrightarrow{\alpha|\psi} C'$, for some $\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. By construction of R we have $(C, \hat{\phi}, t) \in R$. We have already shown that the transfer conditions are satisfied for such pairs in R, and hence we can find a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F})$ such that $(C', [\phi \wedge \varphi'_t]_{\sim_{\mathcal{P}}}, t') \in R$, for all $(t',\varphi'_t) \in \mathcal{T}$, and such that

$$\phi \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t',\varphi'_t) \in \mathcal{T}} \theta(t,\alpha,t') \wedge \varphi'_t \wedge \phi.$$

   Now we show that s can mimic all transitions from t, which we use to conclude that s can mimic the transition from C to C'. Since $fts \models s \overset{\varphi}{\leftrightarrows}_f t$, we can find sets $\mathcal{S}_{t'} \subseteq S \times \mathbb{B}(\mathcal{F})$ for each $(t',\varphi'_t) \in \mathcal{T}$ such that, for all $(s',\varphi'_s) \in \mathcal{S}_{t'}$, we have $fts \models s' \overset{\varphi'_s}{\leftrightarrows}_f t'$ and such that

$$\varphi \wedge \theta(t,\alpha,t') \;\Rightarrow_{\mathcal{P}} \bigvee_{(s',\varphi'_s) \in \mathcal{S}_{t'}} \theta(s,\alpha,s') \wedge \varphi'_s.$$

   Since $(C', [\phi \wedge \varphi'_t]_{\sim_{\mathcal{P}}}, t') \in R$, for all $(t',\varphi'_t) \in \mathcal{T}$, by definition of R this means that $fts \models t' \overset{\varphi'_t}{\leftrightarrows}_f u'$, for some $u' \in C'$. Since $fts \models s' \overset{\varphi'_s}{\leftrightarrows}_f t'$ for all $(s',\varphi'_s) \in \mathcal{S}_{t'}$, we derive $fts \models s' \overset{\varphi'_s \wedge \varphi'_t}{\leftrightarrows}_f u'$ using productwise transitivity. By construction of R we have $(C', [\phi \wedge \varphi'_s \wedge \varphi'_t]_{\sim_{\mathcal{P}}}, s') \in R$. We find that

$$\varphi \wedge \bigvee_{(t',\varphi'_t) \in \mathcal{T}} \theta(t,\alpha,t') \wedge \varphi'_t \wedge \phi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t',\varphi'_t) \in \mathcal{T}} \bigvee_{(s',\varphi'_s) \in \mathcal{S}_{t'}} \theta(s,\alpha,s') \wedge \varphi'_s \wedge \varphi'_t \wedge \phi,$$

and hence that

$$\varphi \wedge \phi \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t',\varphi'_t)\in\mathcal{T}} \bigvee_{(s',\varphi'_s)\in\mathcal{S}_{t'}} \theta(s,\alpha,s') \wedge \varphi'_s \wedge \varphi'_t \wedge \phi,$$

from which we conclude that the transfer condition for this pair is also satisfied.

□

Theorem 5.6. *For each FTS fts, and for all $\phi \in \mathbb{B}(\mathcal{F})$, it holds that $\overset{\phi}{\leftrightarrows}_{bf}$-min$_{\mathtt{FTS}}$(fts) $\overset{\phi}{\leftrightarrows}_{bf}$ fts.*

Proof. Let *fts* = (S, $\mathcal{A}$, θ, s_*) be an FTS, let $\phi \in \mathbb{B}(\mathcal{F})$ be a feature expression, and let $\overset{\phi}{\leftrightarrows}_{bf}$-min$_{\mathtt{FTS}}$(*fks*) = ($S'$, $\mathcal{A}$, θ', s'_*) be the naive divergence-blind stuttering feature quotient for ϕ of *fks*. We define the relation R such that:

$$\begin{aligned} R =& \{(s, [\varphi \wedge \phi]_{\sim_{\mathcal{P}}}, C) \in S \times \widehat{\mathbb{B}}(\mathcal{F}) \times S' \mid \exists t \in C \colon \mathit{fts} \models s \overset{\varphi}{\leftrightarrows}_{bf} t\} \cup \\ & \{(C, [\varphi \wedge \phi]_{\sim_{\mathcal{P}}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S \mid \exists t \in C \colon \mathit{fts} \models s \overset{\varphi}{\leftrightarrows}_{bf} t\} \end{aligned}$$

We have to show that R is a branching feature bisimulation relation on *fts* $\uplus$ $\overset{\phi}{\leftrightarrows}_{bf}$-min$_{\mathtt{FTS}}$(*fts*) such that $(s_*, \hat{\phi}, s'_*) \in R$.

Since $\overset{\phi}{\leftrightarrows}_{bf}$ is reflexive, we find that, for all $s \in S$, $(s, \widehat{\phi}, C) \in R$, where $C \in S'$ is such that $s \in C$. Using that $s_* \in s'_*$ by Definition 5.6, it immediately follows that $(s_*, \hat{\phi}, s'_*) \in R$. Hence, it remains to show that R is a branching feature bisimulation relation.

From the definition of R it immediately follows that it is symmetric, and hence it remains to show that R satisfies the conditions from Definition 4.13.

Consider a tuple $(s, [\varphi \wedge \phi]_{\sim_{\mathcal{P}}}, C) \in S \times \mathbb{B}(\mathcal{F}) \times S'$ such that *fts* $\models s \overset{\varphi}{\leftrightarrows}_{bf} t$ and $t \in C$, for some $t \in S$. Let t be such.

Suppose that $s \xrightarrow{\alpha|\psi} s'$, for some$\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$, and for some $s' \in S$. We first show that this transition can be mimicked by state t for all products satisfying $\varphi \wedge \psi$. Since *fts* $\models s \overset{\varphi}{\leftrightarrows}_{bf} t$, there exists a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ such that *fts* $\models s \overset{\varphi'}{\leftrightarrows}_{bf} t'$ and *fts* $\models s' \overset{\varphi''}{\leftrightarrows}_{bf} t''$, for all $(t', \varphi', t'', \varphi'') \in \mathcal{T}$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t',\varphi',t'',\varphi'')\in\mathcal{T}} \check{\theta}_\tau(t,t') \wedge \varphi' \wedge ((t' = t'' \wedge \alpha = \tau) \vee \theta(t', \alpha, t'')) \wedge \varphi''.$$

Now we show that C can mimic all transitions from t, which we use to conclude that C can mimic the transition from s to s'. Using that $t \in C$, from Definition 5.6 it follows that $\check{\theta}_\tau(t,t') \Rightarrow_{\mathcal{P}} \check{\theta}'_\tau(C, C_{t'})$ and $\theta(t', \alpha, t'') \Rightarrow_{\mathcal{P}} \theta'(C_{t'}, \alpha, C_{t''})$, with $t' \in C_{t'}$ and $t'' \in C_{t''}$, for all $(t', \varphi', t'', \varphi'') \in \mathcal{T}$. Since $t' \in C_{t'}$ and *fts* $\models s \overset{\varphi'}{\leftrightarrows}_{bf} t'$, and $t'' \in C_{t''}$ and *fts* $\models s' \overset{\varphi''}{\leftrightarrows}_{bf} t''$, by construction of R we have $(s, [\varphi' \wedge \phi]_{\sim_{\mathcal{P}}}, C_{t'}) \in R$ and $(s', [\varphi'' \wedge \phi]_{\sim_{\mathcal{P}}}, C_{t''}) \in R$, for all $(t', \varphi', t'', \varphi'') \in \mathcal{T}$. We find that

$$\begin{aligned} \phi \wedge \textstyle\bigvee_{(t',\varphi',t'',\varphi'')\in\mathcal{T}} & \check{\theta}_\tau(t,t') \wedge \varphi' \wedge ((t' = t'' \wedge \alpha = \tau) \vee \theta(t', \alpha, t'')) \wedge \varphi'' \;\Rightarrow_{\mathcal{P}} \\ & \textstyle\bigvee_{(t',\varphi',t'',\varphi'')\in\mathcal{T}} \check{\theta}'_\tau(C, C_{t'}) \wedge \varphi' \wedge ((C_{t'} = C_{t''} \wedge \alpha = \tau) \vee \theta'(C_{t'}, \alpha, C_{t''})) \wedge \varphi'' \wedge \phi, \end{aligned}$$

and hence the transfer condition for this tuple is satisfied.

We prove the transfer conditions for the remaining tuples of R in two steps, using a case distinction.

1. First consider a tuple $(C, \hat{\phi}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \in C$.

   Suppose that $C \xrightarrow{\alpha|\psi} C'$, for some $\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. From Definition 5.6 it follows that $\psi = \bigvee\{\theta(u, \alpha, u') \mid u \in C \wedge u' \in C'\}$. Pick a pair $(u, u') \in C \times C'$. Since both $s \in C$ and $u \in C$, by Definition 5.6 we know that $fts \models s \overset{\phi}{\leftrightarrows}_{bf} u$. Hence we can find a set $\mathcal{S}_{(u,u')} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ for each tuple $(u, u') \in C \times C'$ such that $fts \models s' \overset{\varphi'}{\leftrightarrows}_{bf} u$ and $fks \models s'' \overset{\varphi''}{\leftrightarrows}_{bf} u'$, for all $(s', \varphi', s'', \varphi'') \in \mathcal{S}_{(u,u')}$, and such that

   $$\phi \wedge \theta(u, \alpha, u') \;\Rightarrow_{\mathcal{P}} \bigvee_{(s',\varphi',s'',\varphi'') \in \mathcal{S}_{(u,u')}} \check{\theta}_\tau(s, s') \wedge \varphi' \wedge ((s' = s'' \wedge \alpha = \tau) \vee \theta(s', \alpha, s'')) \wedge \varphi''.$$

   For each tuple $(u, u') \in C \times C'$ we have that $fts \models s' \overset{\varphi'}{\leftrightarrows}_{bf} u$ and $fts \models s'' \overset{\varphi''}{\leftrightarrows}_{bf} u'$, and hence by construction of R we have $(C, [\varphi' \wedge \phi]_{\sim_{\mathcal{P}}}, s') \in R$ and $(C', [\varphi'' \wedge \phi]_{\sim_{\mathcal{P}}}, s'') \in R$, for all $(s', \varphi', s'', \varphi'') \in \mathcal{S}_{(u,u')}$. We find that

   $$\phi \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(u,u') \in C \times C'} \bigvee_{(s',\varphi',s'',\varphi'') \in \mathcal{T}_{(u,u')}} \check{\theta}_\tau(s, s') \wedge \varphi' \wedge ((s' = s'' \wedge \alpha = \tau) \vee \theta(s', \alpha, s'')) \wedge \varphi'' \wedge \phi,$$

   and hence the transfer condition for this tuple is satisfied.

2. Lastly consider a tuple $(C, [\varphi \wedge \phi]_{\sim_{\mathcal{P}}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \notin C$, and such that $fts \models s \overset{\varphi}{\leftrightarrows}_{bf} t$ and $t \in C$, for some $t \in S$. Let t be such.

   Suppose that $C \xrightarrow{\alpha|\psi} C'$, for some $\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. By construction of R we have $(C, \hat{\phi}, t) \in R$. We have already shown that the transfer conditions are satisfied for such pairs in R, and hence we can find a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ such that $(C, [\phi \wedge \varphi'_t]_{\sim_{\mathcal{P}}}, t') \in R$ and $(C', [\phi \wedge \varphi''_t]_{\sim_{\mathcal{P}}}, t'') \in R$, for all $(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}$, and such that

   $$\phi \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t',\varphi'_t,t'',\varphi''_t) \in \mathcal{T}} \check{\theta}_\tau(t, t') \wedge \varphi'_t \wedge ((t' = t'' \wedge \alpha = \tau) \vee \theta(t', \alpha, t'')) \wedge \varphi''_t \wedge \phi.$$

   Now we show that s can mimic all transitions from t, which we use to conclude that s can mimic the transition from C to C'. Since $fts \models s \overset{\varphi}{\leftrightarrows}_{bf} t$, we can find sets $\mathcal{S}_{(t',t'')} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ for each $(t', \varphi'_t, t'', \varphi'') \in \mathcal{T}$ such that, for all $(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{(t',t'')}$, we have $fts \models s' \overset{\varphi'_s}{\leftrightarrows}_{bf} t'$ and $fts \models s'' \overset{\varphi''_s}{\leftrightarrows}_{bf} t''$, and such that

   $$\varphi \wedge \check{\theta}_\tau(t, t') \wedge ((t' = t'' \wedge \alpha = \tau) \vee \theta(t', \alpha, t'')) \;\Rightarrow_{\mathcal{P}} \bigvee_{(s',\varphi'_s,s'',\varphi''_s) \in \mathcal{S}_{t',t''}} \check{\theta}_\tau(s, s') \wedge \varphi'_s \wedge ((s' = s'' \wedge \alpha = \tau) \vee \theta(s', \alpha, s'')) \wedge \varphi''_s.$$

Since $(C, [\phi \wedge \varphi'_t]_{\sim_\mathcal{P}}, t') \in R$ and $(C, [\phi \wedge \varphi''_t]_{\sim_\mathcal{P}}, t'') \in R$, for all $(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}$, by definition of R this means that $\mathit{fts} \models t' \overset{\varphi'_t}{\leftrightarrows}_{bf} u$ and $\mathit{fts} \models t'' \overset{\varphi''_t}{\leftrightarrows}_{bf} u'$, for some $u \in C$, $u', \in C'$. Since $\mathit{fts} \models s' \overset{\varphi'_s}{\leftrightarrows}_{bf} t'$ and $\mathit{fts} \models s'' \overset{\varphi''_s}{\leftrightarrows}_{bf} t''$ for all $(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{(t',t'')}$, we derive $\mathit{fts} \models s' \overset{\varphi'_s \wedge \varphi'_t}{\leftrightarrows}_{bf} u$ and $\mathit{fts} \models s'' \overset{\varphi''_s \wedge \varphi''_t}{\leftrightarrows}_{bf} u'$ using productwise transitivity. By construction of R we have $(C, [\phi \wedge \varphi'_s \wedge \varphi'_t]_{\sim_\mathcal{P}}, s') \in R$ and $(C', [\phi \wedge \varphi''_s \wedge \varphi''_t]_{\sim_\mathcal{P}}, s'') \in R$. We find that

$$\begin{aligned}\varphi \wedge \textstyle\bigvee_{(t',\varphi'_t,t'',\varphi''_t)\in\mathcal{T}} \check{\theta}_\tau(t,t') \wedge \varphi'_t \wedge ((t' = t'' \wedge \alpha = \tau) \vee \theta(t', \alpha, t'')) \wedge \varphi''_t \wedge \phi \;\Rightarrow_\mathcal{P}\\ \textstyle\bigvee_{(t',\varphi'_t,t'',\varphi''_t)\in\mathcal{T}} \bigvee_{(s',\varphi'_s,s'',\varphi''_s)\in\mathcal{S}_{t',t''}} \check{\theta}_\tau(s,s') \wedge \varphi'_s \wedge \varphi'_t \wedge\\ ((s' = s'' \wedge \alpha = \tau) \vee \theta(s', \alpha, s'')) \wedge \varphi''_s \wedge \varphi''_t \wedge \phi,\end{aligned}$$

and hence that

$$\begin{aligned}\varphi \wedge \phi \wedge \psi \;\Rightarrow_\mathcal{P} \textstyle\bigvee_{(t',\varphi'_t,t'',\varphi''_t)\in\mathcal{T}} \bigvee_{(s',\varphi'_s,s'',\varphi''_s)\in\mathcal{S}_{t',t''}} \check{\theta}_\tau(s,s') \wedge \varphi'_s \wedge \varphi'_t \wedge\\ ((s' = s'' \wedge \alpha = \tau) \vee \theta(s', \alpha, s'')) \wedge \varphi''_s \wedge \varphi''_t \wedge \phi,\end{aligned}$$

from which we conclude that the transfer condition for this pair is also satisfied.

□

Theorem 5.7. *For each FTS fts and for all* $\mathit{fts}' \in \leftrightarrows_{cf}\text{-min}_{\mathtt{FTS}}(\mathit{fts})$ *it holds that* $\mathit{fts}' \leftrightarrows_f \mathit{fts}$.

Proof. Let $\mathit{fts} = (S, \mathcal{A}, \theta, s_*)$ be an FTS, let $\phi \in \mathbb{B}(\mathcal{F})$ be a feature expression, and let $\mathit{fts}' = (S', \mathcal{A}, \theta', s'_*)$ be a coherent feature bisimulation quotient of fts. We define the relation R such that:

$$\begin{aligned}R = &\{(s, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_\mathcal{P}}, C) \in S \times \widehat{\mathbb{B}}(\mathcal{F}) \times S' \mid \exists t \in C\colon \mathit{fts} \models s \overset{\varphi}{\leftrightarrows}_f t\} \cup\\ &\{(C, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_\mathcal{P}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S \mid \exists t \in C\colon \mathit{fts} \models s \overset{\varphi}{\leftrightarrows}_f t\}\end{aligned}$$

We have to show that R is a feature bisimulation relation on $\mathit{fts} \uplus \mathit{fts}'$ such that $(s_*, \widehat{\mathsf{true}}, s'_*) \in R$.

Since $\leftrightarrows_f$ is reflexive, we find that, for all $s \in S$, $(s, \widehat{\varrho(s)}, C) \in R$, where $C \in S'$ is such that $s \in C$. Using that $s_* \in s'_*$ by Definition 5.8, it immediately follows that $(s_*, \mathsf{true}, s'_*) \in R$. Hence, it remains to show that R is a feature bisimulation relation.

From the definition of R it immediately follows that it is symmetric, and hence it remains to show that R satisfies the conditions from Definition 4.5.

Consider a tuple $(s, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_\mathcal{P}}, C) \in S \times \mathbb{B}(\mathcal{F}) \times S'$ such that $\mathit{fts} \models s \overset{\varphi}{\leftrightarrows}_f t$ and $t \in C$, for some $t \in S$. Let t be such.

Suppose that $s \xrightarrow{\alpha|\psi} s'$, for some $\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$, and for some $s' \in S$. We first show that this transition can be mimicked by state t for all products satisfying $\varphi \wedge \psi$. Since $\mathit{fts} \models s \overset{\varphi}{\leftrightarrows}_f t$, there exists a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F})$ such that $\mathit{fts} \models s' \overset{\varphi'}{\leftrightarrows}_f t'$, for all $(t', \varphi') \in \mathcal{T}$, and such that

$$\varphi \wedge \psi \;\Rightarrow_\mathcal{P} \bigvee_{(t',\varphi')\in\mathcal{T}} \theta(t, \alpha, t') \wedge \varphi'.$$

Now we show that C can mimic all transitions from t, which we use to conclude that C can mimic the transition from s to s'. Using that $t \in C$, from Definition 5.8 it follows that $(\theta(t,\alpha,t') \wedge \varrho(t)) \Rightarrow_{\mathcal{P}} \theta'(C,\alpha,C_{t'})$, with $t' \in C_{t'}$, for all $(t',\varphi') \in \mathcal{T}$. Since $t' \in C_{t'}$ and $\mathit{fts} \models s' \overset{\varphi'}{\leftrightarrows}_f t'$, by construction of R we have $(s',[\varphi' \wedge \varrho(s') \wedge \varrho(t')]_{\sim_{\mathcal{P}}}, C_{t'}) \in R$, for all $(t',\varphi') \in \mathcal{T}$. Using that $\varrho(s) \wedge \psi \Rightarrow_{\mathcal{P}} \varrho(s')$ and $\varrho(t) \wedge \theta(t,t') \Rightarrow_{\mathcal{P}} \varrho(t')$, for all $(t',\varphi') \in \mathcal{T}$, by definition of reachability, we find that

$$\varrho(s) \wedge \varrho(t) \wedge \bigvee_{(t',\varphi') \in \mathcal{T}} \theta(t,\alpha,t') \wedge \varphi' \Rightarrow_{\mathcal{P}} \bigvee_{(t',\varphi') \in \mathcal{T}} \theta'(C,\alpha,C_{t'}) \wedge \varphi' \wedge \varrho(s') \wedge \varrho(t'),$$

and hence the transfer condition for this tuple is satisfied.

We prove the transfer conditions for the remaining tuples of R in two steps, using a case distinction.

1. First consider a tuple $(C, \widehat{\varrho(s)}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \in C$.

   Suppose that $C \xrightarrow{\alpha|\psi} C'$, for some $\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. From Definition 5.8 it follows that $\psi = \bigvee\{\theta(u,\alpha,u') \wedge \varrho(u) \mid u \in C \wedge u' \in C'\}$. Pick a pair $(u,u') \in C \times C'$. Since both $s \in C$ and $u \in C$, by Definition 5.8 we know that $\mathit{fks} \models s \leftrightarrows_{cf} u$. Hence we can find a set $\mathcal{S}_{(u,u')} \subseteq S \times \mathbb{B}(\mathcal{F})$ for each tuple $(u,u') \in C \times C'$ such that $\mathit{fts} \models s' \overset{\varphi'}{\leftrightarrows}_f u'$, for all $(s',\varphi') \in \mathcal{S}_{(u,u')}$, and such that

   $$\varrho(s) \wedge \varrho(u) \wedge \theta(u,\alpha,u') \Rightarrow_{\mathcal{P}} \bigvee_{(s',\varphi') \in \mathcal{S}_{(u,u')}} \theta(s,\alpha,s') \wedge \varphi'.$$

   For each tuple $(u,u') \in C \times C'$ we have that $\mathit{fts} \models s' \overset{\varphi'}{\leftrightarrows}_f u'$, and hence by construction of R we have $(C',[\varphi' \wedge \varrho(s') \wedge \varrho(u')]_{\sim_{\mathcal{P}}}, s') \in R$, for all $(s',\varphi') \in \mathcal{S}_{(u,u')}$. Using that $\varrho(s) \wedge \theta(s,\alpha,s') \Rightarrow_{\mathcal{P}} \varrho(s')$ and $\varrho(u) \wedge \theta(u,\alpha,u') \Rightarrow_{\mathcal{P}} \varrho(u')$, we find that

   $$\varrho(s) \wedge \psi \Rightarrow_{\mathcal{P}} \bigvee_{(u,u') \in C \times C'} \bigvee_{(s',\varphi') \in \mathcal{T}_{(u,u')}} \theta(s,\alpha,s') \wedge \varphi' \wedge \varrho(s') \wedge \varrho(u'),$$

   and hence the transfer condition for this tuple is satisfied.

2. Lastly consider a tuple $(C, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_{\mathcal{P}}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \notin C$, and such that $\mathit{fts} \models s \overset{\varphi}{\leftrightarrows}_f t$ and $t \in C$, for some $t \in S$. Let t be such.

   Suppose that $C \xrightarrow{\alpha|\psi} C'$, for some $\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. By construction of R we have $(C, \widehat{\varrho(t)}, t) \in R$. We have already shown that the transfer conditions are satisfied for such pairs in R, and hence we can find a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F})$ such that $(C', \hat{\varphi}'_t, t') \in R$, for all $(t',\varphi'_t) \in \mathcal{T}$, and such that

   $$\varrho(t) \wedge \psi \Rightarrow_{\mathcal{P}} \bigvee_{(t',\varphi'_t) \in \mathcal{T}} \theta(t,\alpha,t') \wedge \varphi'_t.$$

Now we show that s can mimic all transitions from t, which we use to conclude that s can mimic the transition from C to C'. Since $fts \models s \overset{\varphi}{\leftrightarrows}_f t$, we can find sets $\mathcal{S}_{t'} \subseteq S \times \mathbb{B}(\mathcal{F})$ for each $(t', \varphi'_t) \in \mathcal{T}$ such that, for all $(s', \varphi'_s) \in \mathcal{S}_{t'}$, we have $fts \models s' \overset{\varphi'_s}{\leftrightarrows}_f t'$ and such that

$$\varphi \wedge \theta(t, \alpha, t') \;\Rightarrow_{\mathcal{P}} \bigvee_{(s', \varphi'_s) \in \mathcal{S}_{t'}} \theta(s, \alpha, s') \wedge \varphi'_s.$$

Since $(C', \hat{\varphi}'_t, t') \in R$, for all $(t', \varphi'_t) \in \mathcal{T}$, by definition of R this means that $\varphi'_t \sim_{\mathcal{P}} \varphi''_t \wedge \varrho(t') \wedge \varrho(u_{t'})$, for some $\varphi''_t \in \mathbb{B}(\mathcal{F})$ and some $u_{t'} \in C'$, such that $fts \models t' \overset{\varphi''_t}{\leftrightarrows}_f u_{t'}$. Since $fts \models s' \overset{\varphi'_s}{\leftrightarrows}_f t'$ for all $(s', \varphi'_s) \in \mathcal{S}_{t'}$, we derive $fts \models s' \overset{\varphi'_s \wedge \varphi''_t}{\leftrightarrows}_f u_{t'}$ using productwise transitivity. By construction of R we have $(C', [\varphi'_s \wedge \varphi''_t \wedge \varrho(s') \wedge \varrho(u_{t'})]_{\sim_{\mathcal{P}}}, s') \in R$. Using that $\varrho(s) \wedge \psi \Rightarrow_{\mathcal{P}} \varrho(s')$ and $\varrho'_t \Rightarrow_{\mathcal{P}} \varrho''_t \wedge \varrho(u_{t'})$, we find that We find that

$$\varphi \wedge \bigvee_{(t', \varphi'_t) \in \mathcal{T}} \theta(t, \alpha, t') \wedge \varphi'_t \;\Rightarrow_{\mathcal{P}} \bigvee_{(t', \varphi'_t) \in \mathcal{T}} \bigvee_{(s', \varphi'_s) \in \mathcal{S}_{t'}} \theta(s, \alpha, s') \wedge \varphi'_s \wedge \varphi'_t,$$

and hence that

$$\varphi \wedge \varrho(s) \wedge \varrho(t) \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t', \varphi'_t) \in \mathcal{T}} \bigvee_{(s', \varphi'_s) \in \mathcal{S}_{t'}} \theta(s, \alpha, s') \wedge \varphi'_s \wedge \varphi''_t \wedge \varrho(s') \wedge \varrho(u_{t'}),$$

from which we conclude that the transfer condition for this pair is also satisfied.

□

Theorem 5.8. *For each FTS fts and for all* $fts' \in \leftrightarrows_{cbf}\text{-min}_{\mathtt{FTS}}(fts)$ *it holds that* $fts' \leftrightarrows_{bf} fts$.

Proof. Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, let $\phi \in \mathbb{B}(\mathcal{F})$ be a feature expression, and let $fts' = (S', \mathcal{A}, \theta', s'_*)$ be a coherent branching feature quotient of fts. We define the relation R such that:

$$\begin{aligned} R = &\{(s, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_{\mathcal{P}}}, C) \in S \times \widehat{\mathbb{B}}(\mathcal{F}) \times S' \mid \exists t \in C \colon fts \models s \overset{\varphi}{\leftrightarrows}_{bf} t\} \cup \\ &\{(C, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_{\mathcal{P}}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S \mid \exists t \in C \colon fts \models s \overset{\varphi}{\leftrightarrows}_{bf} t\} \end{aligned}$$

We have to show that R is a branching feature bisimulation relation on $fts \uplus fts'$ such that $(s_*, \widehat{\mathsf{true}}, s'_*) \in R$.

Since $\leftrightarrows_{bf}$ is reflexive, we find that, for all $s \in S$, $(s, \widehat{\varrho(s)}, C) \in R$, where $C \in S'$ is such that $s \in C$. Using that $s_* \in s'_*$ by Definition 5.8, it immediately follows that $(s_*, \widehat{\mathsf{true}}, s'_*) \in R$. Hence, it remains to show that R is a branching feature bisimulation relation.

From the definition of R it immediately follows that it is symmetric, and hence it remains to show that R satisfies the conditions from Definition 4.13.

Consider a tuple $(s, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_{\mathcal{P}}}, C) \in S \times \mathbb{B}(\mathcal{F}) \times S'$ such that $fts \models s \overset{\varphi}{\leftrightarrows}_{bf} t$ and $t \in C$, for some $t \in S$. Let t be such.

Suppose that $s \xrightarrow{\alpha|\psi} s'$, for some $\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$, and for some $s' \in S$. We first show that this transition can be mimicked by state t for all products satisfying $\varphi \wedge \psi$. Since $\mathit{fts} \models s \overset{\varphi}{\leftrightarrows}_{bf} t$, there exists a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ such that $\mathit{fts} \models s \overset{\varphi'}{\leftrightarrows}_{bf} t'$ and $\mathit{fts} \models s' \overset{\varphi''}{\leftrightarrows}_{bf} t''$, for all $(t', \varphi', t'', \varphi'') \in \mathcal{T}$, and such that

$$\varphi \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t',\varphi',t'',\varphi'') \in \mathcal{T}} \check{\theta}_\tau(t,t') \wedge \varphi' \wedge ((t' = t'' \wedge \alpha = \tau) \vee \theta(t', \alpha, t'')) \wedge \varphi''.$$

Now we show that C can mimic all transitions from t, which we use to conclude that C can mimic the transition from s to s'. Using that $t \in C$, from Definition 5.8 it follows that $(\check{\theta}_\tau(t,t') \wedge \varrho(t)) \Rightarrow_{\mathcal{P}} \check{\theta}'_\tau(C, C_{t'})$ and $(\theta(t', \alpha, t'') \wedge \varrho(t')) \Rightarrow_{\mathcal{P}} \theta'(C_{t'}, \alpha, C_{t''})$, with $t' \in C_{t'}$ and $t'' \in C_{t''}$, for all $(t', \varphi', t'', \varphi'') \in \mathcal{T}$. Since $t' \in C_{t'}$ and $\mathit{fts} \models s \overset{\varphi'}{\leftrightarrows}_{bf} t'$, and $t'' \in C_{t''}$ and $\mathit{fts} \models s' \overset{\varphi''}{\leftrightarrows}_{bf} t''$, by construction of R we have $(s, [\varphi' \wedge \varrho(s) \wedge \varrho(t')]_{\sim_{\mathcal{P}}}, C_{t'}) \in R$ and $(s', [\varphi'' \wedge \varrho(s') \wedge \varrho(t'')]_{\sim_{\mathcal{P}}}, C_{t''}) \in R$, for all $(t', \varphi', t'', \varphi'') \in \mathcal{T}$. Using that $\varrho(s) \wedge \psi \Rightarrow_{\mathcal{P}} \varrho(s')$ and $\varrho(t) \wedge \check{\theta}_\tau(t,t') \wedge ((t' = t'' \wedge \alpha = \tau) \vee \theta(t', \alpha, t'')) \Rightarrow_{\mathcal{P}} \varrho(t) \wedge \varrho(t')$, for all $(t', \varphi', t'', \varphi'') \in \mathcal{T}$, by definition of reachability, we find that

$$\begin{aligned} &\varrho(s) \wedge \varrho(t) \;\wedge \textstyle\bigvee_{(t',\varphi',t'',\varphi'') \in \mathcal{T}} \check{\theta}_\tau(t,t') \wedge \varphi' \wedge ((t' = t'' \wedge \alpha = \tau) \vee \theta(t', \alpha, t'')) \wedge \varphi'' \;\Rightarrow_{\mathcal{P}} \\ &\quad \textstyle\bigvee_{(t',\varphi',t'',\varphi'') \in \mathcal{T}} \check{\theta}'_\tau(C, C_{t'}) \wedge \varphi' \wedge \varrho(s) \wedge \varrho(t') \wedge \\ &\qquad ((C_{t'} = C_{t''} \wedge \alpha = \tau) \vee \theta'(C_{t'}, \alpha, C_{t''})) \wedge \varphi'' \;\wedge\; \varrho(s') \wedge \varrho(t''), \end{aligned}$$

and hence the transfer condition for this tuple is satisfied.

We prove the transfer conditions for the remaining tuples of R in two steps, using a case distinction.

1. First consider a tuple $(C, \widehat{\varrho(s)}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \in C$.

   Suppose that $C \xrightarrow{\alpha|\psi} C'$, for some $\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. From Definition 5.8 it follows that $\psi = \bigvee\{\theta(u, \alpha, u') \wedge \varrho(u) \mid u \in C \wedge u' \in C'\}$. Pick a pair $(u, u') \in C \times C'$. Since both $s \in C$ and $u \in C$, by Definition 5.8 we know that $\mathit{fts} \models s \leftrightarrows_{cbf} u$. Hence we can find a set $\mathcal{S}_{(u,u')} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ for each tuple $(u, u') \in C \times C'$ such that $\mathit{fts} \models s' \overset{\varphi'}{\leftrightarrows}_{bf} u$ and $\mathit{fts} \models s'' \overset{\varphi''}{\leftrightarrows}_{bf} u'$, for all $(s', \varphi', s'', \varphi'') \in \mathcal{S}_{(u,u')}$, and such that

   $$\varrho(s) \wedge \varrho(u) \wedge \theta(u, \alpha, u') \;\Rightarrow_{\mathcal{P}} \bigvee_{(s',\varphi',s'',\varphi'') \in \mathcal{S}_{(u,u')}} \check{\theta}_\tau(s,s') \wedge \varphi' \wedge ((s' = s'' \wedge \alpha = \tau) \vee \theta(s', \alpha, s'')) \wedge \varphi''.$$

   For each tuple $(u, u') \in C \times C'$ we have that $\mathit{fts} \models s' \overset{\varphi'}{\leftrightarrows}_{bf} u$ and $\mathit{fts} \models s'' \overset{\varphi''}{\leftrightarrows}_{bf} u'$, and hence by construction of R we have $(C, [\varphi' \wedge \varrho(s') \wedge \varrho(u)]_{\sim_{\mathcal{P}}}, s') \in R$ and $(C', [\varphi'' \wedge \varrho(s'') \wedge \varrho(u')]_{\sim_{\mathcal{P}}}, s'') \in R$, for all $(s', \varphi', s'', \varphi'') \in \mathcal{S}_{(u,u')}$. Using that $\varrho(s) \wedge \check{\theta}_\tau(s,s') \wedge ((s' = s'' \wedge \alpha = \tau) \vee \theta(s', \alpha, s'')) \Rightarrow_{\mathcal{P}} \varrho(s') \wedge \varrho(s'')$ and $\varrho(u) \wedge \theta(u, \alpha, u') \Rightarrow_{\mathcal{P}} \varrho(u')$, we find that

   $$\begin{aligned} &\varrho(s) \wedge \psi \;\Rightarrow_{\mathcal{P}} \textstyle\bigvee_{(u,u') \in C \times C'} \bigvee_{(s',\varphi',s'',\varphi'') \in \mathcal{T}_{(u,u')}} \check{\theta}_\tau(s,s') \wedge \varphi' \wedge \varrho(s') \wedge \varrho(u) \wedge \\ &\qquad ((s' = s'' \wedge \alpha = \tau) \vee \theta(s', \alpha, s'')) \wedge \varphi'' \wedge \varrho(s'') \wedge \varrho(u'), \end{aligned}$$

and hence the transfer condition for this tuple is satisfied.

2. Lastly consider a tuple $(C, [\varphi \wedge \varrho(s) \wedge \varrho(t)]_{\sim_{\mathcal{P}}}, s) \in S' \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ such that $s \notin C$, and such that $\mathit{fts} \models s \overset{\varphi}{\leftrightarrows}_{bf} t$ and $t \in C$, for some $t \in S$. Let t be such.

   Suppose that $C \xrightarrow{\alpha|\psi} C'$, for some $\alpha \in \mathcal{A}_\tau$, $\psi \in \mathbb{B}(\mathcal{F})$, and for some $C' \in S'$. By construction of R we have $(C, \widehat{\varrho(t)}, t) \in R$. We have already shown that the transfer conditions are satisfied for such pairs in R, and hence we can find a set $\mathcal{T} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ such that $(C, \hat{\varphi}'_t, t') \in R$ and $(C', \hat{\varphi}''_t, t'') \in R$, for all $(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}$, and such that

$$\varrho(t) \wedge \psi \;\Rightarrow_{\mathcal{P}} \bigvee_{(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}} \check{\theta}_\tau(t, t') \wedge \varphi'_t \wedge ((t' = t'' \wedge \alpha = \tau) \vee \theta(t', \alpha, t'')) \wedge \varphi''_t.$$

   Now we show that s can mimic all transitions from t, which we use to conclude that s can mimic the transition from C to C'. Since $\mathit{fts} \models s \overset{\varphi}{\leftrightarrows}_{bf} t$, we can find sets $\mathcal{S}_{(t', t'')} \subseteq S \times \mathbb{B}(\mathcal{F}) \times S \times \mathbb{B}(\mathcal{F})$ for each $(t', \varphi'_t, t'', \varphi'') \in \mathcal{T}$ such that, for all $(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{(t', t'')}$, we have $\mathit{fts} \models s' \overset{\varphi'_s}{\leftrightarrows}_{bf} t'$ and $\mathit{fts} \models s'' \overset{\varphi''_s}{\leftrightarrows}_{bf} t''$, and such that

$$\begin{aligned} &\varphi \wedge \check{\theta}_\tau(t, t') \wedge ((t' = t'' \wedge \alpha = \tau) \vee \theta(t', \alpha, t'')) \;\Rightarrow_{\mathcal{P}} \\ &\quad \textstyle\bigvee_{(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{t', t''}} \check{\theta}_\tau(s, s') \wedge \varphi'_s \wedge ((s' = s'' \wedge \alpha = \tau) \vee \theta(s', \alpha, s'')) \wedge \varphi''_s. \end{aligned}$$

   Since $(C, \hat{\varphi}'_t, t') \in R$ and $(C', \hat{\varphi}''_t, t'') \in R$, for all $(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}$, by definition of R this means that $\varphi'_t \sim_{\mathcal{P}} \varphi^*_t \wedge \varrho(t') \wedge \varrho(u_{t'})$ and $\varphi''_t \sim_{\mathcal{P}} \varphi^{**}_t \wedge \varrho(t'') \wedge \varrho(u_{t''})$, for some $\varphi^*_t, \varphi^{**}_t \in \mathbb{B}(\mathcal{F})$ and some $u_{t'} \in C$, $u_{t''} \in C'$, such that $\mathit{fts} \models t' \overset{\varphi^*_t}{\leftrightarrows}_{bf} u_{t'}$ and $\mathit{fts} \models t'' \overset{\varphi^{**}_t}{\leftrightarrows}_{bf} u_{t''}$. Since $\mathit{fts} \models s' \overset{\varphi'_s}{\leftrightarrows}_{bf} t'$ and $\mathit{fts} \models s'' \overset{\varphi''_s}{\leftrightarrows}_{bf} t''$ for all $(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{(t', t'')}$, we derive $\mathit{fts} \models s' \overset{\varphi'_s \wedge \varphi^*_t}{\leftrightarrows}_{bf} u_{t'}$ and $\mathit{fts} \models s'' \overset{\varphi''_s \wedge \varphi^{**}_t}{\leftrightarrows}_{bf} u_{t''}$ using productwise transitivity. By construction of R we have $(C, [\varphi'_s \wedge \varphi^*_t \wedge \varrho(s') \wedge \varrho(u_{t'})]_{\sim_{\mathcal{P}}}, s') \in R$ and $(C', [\varphi''_s \wedge \varphi^{**}_t \wedge \varrho(s'') \wedge \varrho(u_{t''})]_{\sim_{\mathcal{P}}}, s'') \in R$. Using that $\varrho(s) \wedge \check{\theta}_\tau(s, s') \wedge ((s' = s'' \wedge \alpha = \tau) \vee \theta(s', s'')) \Rightarrow_{\mathcal{P}} \varrho(s') \wedge \varrho(s'')$ and $\varphi'_t \Rightarrow_{\mathcal{P}} \varphi^*_t \wedge \varrho(u_{t'})$ and $\varphi''_t \Rightarrow_{\mathcal{P}} \varphi^{**}_t \wedge \varrho(u_{t''})$, we find that

$$\begin{aligned} &\varphi \wedge \textstyle\bigvee_{(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}} \check{\theta}_\tau(t, t') \wedge \varphi'_t \wedge ((t' = t'' \wedge \alpha = \tau) \vee \theta(t', \alpha, t'')) \wedge \varphi''_t \;\Rightarrow_{\mathcal{P}} \\ &\quad \textstyle\bigvee_{(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}} \bigvee_{(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{t', t''}} \check{\theta}_\tau(s, s') \wedge \varphi'_s \wedge \varphi'_t \wedge \\ &\qquad ((s' = s'' \wedge \alpha = \tau) \vee \theta(s', \alpha, s'')) \wedge \varphi''_s \wedge \varphi''_t, \end{aligned}$$

   and hence that

$$\begin{aligned} &\varphi \wedge \varrho(s) \wedge \varrho(t) \wedge \psi \;\Rightarrow_{\mathcal{P}} \\ &\quad \textstyle\bigvee_{(t', \varphi'_t, t'', \varphi''_t) \in \mathcal{T}} \bigvee_{(s', \varphi'_s, s'', \varphi''_s) \in \mathcal{S}_{t', t''}} \check{\theta}_\tau(s, s') \wedge \varphi'_s \wedge \varphi^*_t \wedge \varrho(s') \wedge \varrho(u_{t'}) \wedge \\ &\qquad ((s' = s'' \wedge \alpha = \tau) \vee \theta(s', \alpha, s'')) \wedge \varphi''_s \wedge \varphi^{**}_t \wedge \varrho(s'') \wedge \varrho(u_{t''}), \end{aligned}$$

   from which we conclude that the transfer condition for this pair is also satisfied.

□

Appendix D

Supplementaries for Chapter 6

This appendix contains the full proofs that were omitted in Chapter 6 (Section D.1). It furthermore contains formal descriptions of the data types used in the algorithm (Section D.2), as well as pseudo code of the used subroutines (Section D.3).

D.1 Full proofs for Chapter 6

Lemma 6.1. *Let the graph $\mathcal{G} = (V, E)$ and the FTS $fts_G = (S_G, \mathcal{A}_G, \theta_G, s_{*G})$ be as given above. For distinct $s, t \in S_G$, we have that*

$$fts_G \models s \leftrightarrows_{cf} t \Leftrightarrow (\exists u, v \in V : s = s_u \wedge t = s_v \wedge (u, v) \notin E).$$

Proof. We prove the lemma in two steps.

1. Assume $fts_G \models s \leftrightarrows_{cf} t$. We first show that $\exists u, v \in V : s = s_u \wedge t = s_v$. Recall that $S_G = \{s_*, s_\perp\} \cup \{ s_v \mid v \in V \}$, for distinguished states s_* and $s_\perp$. Assume $s = s_*$. Pick a product $P_w \in \mathcal{P}$ such that $P_w \models \varrho(s) \wedge \varrho(t)$, for some $w \in V$. Such a product exists since $\varrho(s_*) \sim_{\mathcal{P}} \mathsf{true}$, and $\varrho(t) \not\sim_{\mathcal{P}} \mathsf{false}$, by construction of fts_G. We find that $s \xrightarrow{a} s_w \xrightarrow{a} s_\perp$ in $fts_G|_{P_w}$, while $t \xrightarrow{a} s_\perp$ or $\nexists t' \in S_G : t \xrightarrow{a} t'$, by definition of θ_G. It follows that $fts_G \not\models s \leftrightarrows_{cf} t$, and hence we conclude $s \neq s_*$. Similarly, $t \neq s_*$.

   Assume $s = s_\perp$. By definition of θ_G there exists a product $P \in \mathcal{P}$ such that $t \xrightarrow{a} t'$ in $fts_G|_P$, for some $t' \in S_G$. Since s does not have any outgoing transitions, it follows that $fts_G \not\models s \leftrightarrows_{cf} t$, and hence we conclude $s \neq s_\perp$. Similarly, $t \neq s_\perp$. It follows that indeed $\exists u, v \in V : s = s_u \wedge t = s_v$. Let u and v be such.

   We now show that $(u, v) \notin E$. Assume $(u, v) \in E$. By definition of θ_G we have $P_u \models \varrho(s) \wedge \varrho(t)$. Furthermore we have $s \xrightarrow{a} s_\perp$ and $\nexists t' \in S_G : t \xrightarrow{a} s_\perp$ in $fts_G|_{P_u}$. It follows that $fts_G \not\models s \leftrightarrows_{cf} t$, and hence we conclude$(u, v) \notin E$.

2. Assume $\exists u, v \in V : s = s_u \wedge t = s_v \wedge (u, v) \notin E$. By definition of θ_G we have $P_u \not\models \varrho(v)$ and $P_v \not\models \varrho(u)$. Furthermore both s and t have only one outgoing transition: $s \xrightarrow{a|P_u} s_\perp$ and $t \xrightarrow{a|P_v} t_\perp$. It immediately folllows that indeed $fts_G \models s \leftrightarrows_{cf} t$.

□

D.2 Data type definitions

Block {
  list<**State**> states
  list<**Transition**> inert
  list<**Transition**> nonInert
  list<(**Action**, **Set**<**Product**>)> labels
}

State {
  list<**Transition**> transitions
  list<**Block**> blocks
  set<**Product**> products
  set<**Product**> flag
}

Transition {
  State from
  State to
  Action action
  set<**Product**> products
}

D.3 Pseudo code of subroutines

Algorithm *RaiseFlags*(tr)
1. **for** $b \in tr.from.blocks$
2. **list** $T := \{tr\}$
3. **for** $t \in T$
4. **set** $newFlag := t.products \cup t.from.flag$
5. **if** $t.from.flag \neq newFlag$
6. **then** $t.from.flag := newFlag$
7. $T := T \cup \{t' \in t.from.transitions \mid t'.label == \tau \wedge b \in t'.from.blocks\}$
8. **return**

Algorithm *LowerFlags*(BL)
1. **for** $b \in BL, s \in b.states$
2. **do** $s.flag := \emptyset$
3. **return**

Algorithm *RemoveBlock*(B)
1. **for** $s \in B.states$
2. $s.blocks.remove(B)$
3. **if** $B' \in toBeProcessed$

4. **then** $toBeProcessed.remove(B)$
5. **if** $B' \in Stable$
6. **then** $Stable.remove(B)$
7. **return**

Algorithm $SplitBlock(B, S)$
1. **list** $inert := \{t \in \bigcup_{s \in S} s.transitions \mid t.label == \tau \wedge t.from \in S\}$
2. **list** $nonInert := \{t \in \bigcup_{s \in S} s.transitions \mid t.label \neq \tau \vee t.from \notin S \vee t.from.blocks \neq \emptyset\}$
3. **list** $labels := \{(t.label, t.products) \mid t \in nonInert\}$
4. **return** (S, $inert$, $nonInert$, $labels$)

Appendix E

Full proofs for Chapter 7

This appendix contains the full proofs that were omitted in Chapter 7.

Lemma 7.1. *Let fks be an FKS. Then, for all $P \in \mathcal{P}$, we have* $\texttt{fts}(\mathit{fks})|_P = \texttt{lts}(\mathit{fks}|_P)$.

Proof. Let $\mathit{fks} = (S, AP, \theta, L, s_*)$ be an FKS and let $P \in \mathcal{P}$ be a product. Let the LTS $\texttt{fts}(\mathit{fks})|_P = (S', \mathcal{A}, \rightarrow, s'_*)$ be the projection to P of its corresponding FTS, and let $\texttt{lts}(\mathit{fks}|_P) = (S'', \mathcal{A}', \rightarrow', s''_*)$ be the corresponding LTS of its projection to P. We establish the isomorphism by proving that there are isomorphisms between S' and S'', $\mathcal{A}$ and $\mathcal{A}'$, $\rightarrow$ and $\rightarrow'$, and s'_* and s''_*.

- The set of states is unchanged under projection, and hence $S' = S \cup \{\bar{s} \mid s \in S\} = S''$.
- The set of actions is unchanged under projection, as well as the set of atomic propositions, and hence $\mathcal{A} = 2^{AP} \cup \{\bot\} = \mathcal{A}'$.
- We have $\rightarrow \; = \{(s, \bot, \bar{s}) \mid s \in S\} \cup \{(\bar{t}, L(t), t) \mid t \in S\} \cup \{(s, \tau, t) \mid P \models \theta(s,t) \wedge L(s) = L(t)\} \cup \{(s, L(t), t) \mid P \models \theta(s,t) \wedge L(s) \neq L(t)\} = \rightarrow'$
- The initial state is unchanged by both projection and the embedding $\texttt{fts}$, and hence $s'_* = s_* = s''_*$.

□

Lemma 7.2. *Let $\mathit{fks} = (S, AP, \theta, L, s_*, \mathcal{F}, \mathcal{P})$ be an FKS. Then we have, for all $s, s' \in S$, $\texttt{fts}^+(\mathit{fks}) \models s \overset{\varphi}{\leftrightarrows}_f s' \Leftrightarrow \texttt{fts}^+(\mathit{fks}) \models \bar{s} \overset{\varphi}{\leftrightarrows}_f \bar{s}'$, for all $\varphi \in \mathbb{B}(\mathcal{F})$.*

Proof. We distinguish two cases.

1. $\hat{\varphi} = \widehat{\mathsf{false}}$. This case is satisfied immediately since we trivially have $\texttt{fts}^+(\mathit{fks}) \models t \overset{\mathsf{false}}{\longleftrightarrow}_f t'$, for all states t, t' of $\texttt{fts}^+(\mathit{fks})$.
2. $\hat{\varphi} \neq \widehat{\mathsf{false}}$.

   Suppose that $\texttt{fts}^+(\mathit{fks}) \models s \overset{\varphi}{\leftrightarrows}_f s'$. By definition of $\texttt{fts}^+$, we know that both s and s' have only one outgoing $\bot$-transition: $s \xrightarrow{\bot|\mathsf{true}} \bar{s}$ and $s' \xrightarrow{\bot|\mathsf{true}} \bar{s}'$. Hence we must have $\texttt{fts}^+(\mathit{fks}) \models \bar{s} \overset{\varphi}{\leftrightarrows}_f \bar{s}'$ in order to satisfy the transfer conditions.

Suppose that $\texttt{fts}^+(\mathit{fks}) \models \bar{s} \overset{\varphi}{\leftrightarrows}_f \bar{s}'$. By definition of $\texttt{fts}^+$, we know that both $\bar{s}$ and $\bar{s}'$ have only one outgoing transition: $\bar{s} \xrightarrow{L(s)|\mathsf{true}} s$ and $\bar{s}' \xrightarrow{L(s')|\mathsf{true}} s'$. Hence we must have $\texttt{fts}^+(\mathit{fks}) \models s \overset{\varphi}{\leftrightarrows}_f s'$ in order to satisfy the transfer conditions.

□

Lemma 7.3. *Let fks be an FKS. Then, for all $P \in \mathcal{P}$, we have $\texttt{fts}^+(\mathit{fks})|_P = \texttt{lts}(\mathit{fks}|_P)$.*

Proof. Let $\mathit{fks} = (S, AP, \theta, L, s_*, \mathcal{F}, \mathcal{P})$ be an FKS and let $P \in \mathcal{P}$ be a product. Let the LTS $\texttt{fts}^+(\mathit{fks})|_P = (S', \mathcal{A}, \rightarrow, s'_*)$ be the projection to P of its corresponding FTS, and let $\texttt{lts}(\mathit{fks}|_P) = (S'', \mathcal{A}', \rightarrow', s''_*)$ be the corresponding LTS of its projection to P. We establish the isomorphism by proving that there are isomorphisms between S' and S'', $\mathcal{A}$ and $\mathcal{A}'$, $\rightarrow$ and $\rightarrow'$, s'_* and s''_*.

- The set of states is unchanged under projection, and hence $S' = S \cup \{\bar{s} \mid s \in S\} = S''$.
- The set of actions is unchanged under projection, as well as the set of atomic propositions, and hence $\mathcal{A} = 2^{AP} \cup \{\bot\} = \mathcal{A}'$.
- We have $\rightarrow \; = \{(s, \bot, \bar{s}) \mid s \in S\} \cup \{(\bar{t}, L(t), t) \mid t \in S\} \cup \{(s, \tau, t) \mid P \models \theta(s,t) \wedge L(s) = L(t)\} \cup \{(s, L(t), t) \mid P \models \theta(s,t) \wedge L(s) \neq L(t)\} = \rightarrow'$
- The initial state is unchanged by both projection and the embedding $\texttt{fts}$, and hence $s'_* = s_* = s''_*$.

□

Lemma 7.4. *Let $\mathit{fks} = (S, AP, \theta, L, s_*, \mathcal{F}, \mathcal{P})$ be an FKS. Then we have, for all $s, s' \in S$, $\texttt{fts}^+(\mathit{fks})|_{\{f_\bot\}} \models s \leftrightarrows s' \Leftrightarrow L(s) = L(s')$.*

Proof. Let $\mathit{fks} = (S, AP, \theta, L, s_*, \mathcal{F}, \mathcal{P})$ be an FKS, and let $s, s' \in S$.

Suppose that $\texttt{fts}^+(\mathit{fks})|_{\{f_\bot\}} \models s \leftrightarrows s'$. By Lemma 7.2 we derive that $\texttt{fts}^+(\mathit{fks})|_{\{f_\bot\}} \models \bar{s} \leftrightarrows \bar{s}'$. By definition of $\texttt{fts}^+$, we know that both $\bar{s}$ and $\bar{s}'$ have only one outgoing transition in $\{f_\bot\}$: $\bar{s} \xrightarrow{L(s)} s$ and $\bar{s}' \xrightarrow{L(s')} s'$. It follows that $L(s) = L(s')$.

Suppose that $L(s) = L(s')$. We show that the relation $R = \{(s, s'), (\bar{s}, \bar{s}')\}$ is a bisimulation relation on $\texttt{fts}^+(\mathit{fks})|_{\{f_\bot\}}$. By definition of $\texttt{fts}^+$, we know that both s and s' have only one outgoing transition in $\texttt{fts}^+(\mathit{fks})|_{\{f_\bot\}}$: $s \xrightarrow{\bot} \bar{s}$ and $s' \xrightarrow{\bot} \bar{s}'$. This is the case since all other outgoing transitions of s and s' in $\texttt{fts}^+(\mathit{fks})$ have feature constraint $\neg f_\bot$ attached to them. Since the pair $(\bar{s}, \bar{s}')$ is included in R, the transfer condition for the pair (s, s') is satisfied. Again by definition of $\texttt{fts}^+$, we know that both $\bar{s}$ and $\bar{s}'$ have only one outgoing transition in $\texttt{fts}^+(\mathit{fks})|_{\{f_\bot\}}$: $\bar{s} \xrightarrow{L(s)} s$ and $\bar{s}' \xrightarrow{L(s')} s'$. Since we know that $L(s) = L(s')$ and the pair (s, s') is included in R, the transfer condition for the pair $(\bar{s}, \bar{s}')$ is satisfied. We can conclude that $\texttt{fts}^+(\mathit{fks})|_{\{f_\bot\}} \models s \leftrightarrows s'$. □

Theorem 7.3. *Let $\mathit{fks} = (S, AP, \theta, L, s_*, \mathcal{F}, \mathcal{P})$ be an FKS.Then, for all $s, s' \in S$, and $\varphi \in \mathbb{B}(\mathcal{F})$, we have $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f s'$ if and only if $\texttt{fts}^+(\mathit{fks}) \models s \overset{\varphi \vee f_\bot}{\leftrightarrows}_f s'$.*

Proof. Let $fks = (S, AP, \theta, L, s_*, \mathcal{F}, \mathcal{P})$ be an FKS, and let $s, s' \in S$ and $\varphi \in \mathbb{B}(\mathcal{F})$.

Suppose $fks \models s \overset{\varphi}{\leftrightarrows}_f s'$. By Theorem 4.1 this implies $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fks|_P \models s \leftrightarrows s'$. By Theorem 7.1 we get $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathtt{lts}(fks|_P) \models s \leftrightarrows s'$, and by Lemma 7.3 this is equivalent to $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathtt{fts}^+(fks)|_P \models s \leftrightarrows s'$. Now we use Theorem 4.6 to obtain $\mathtt{fts}^+(fks) \models s \overset{\varphi}{\leftrightarrows}_f s'$. By Definition 4.2 we have $L(s) = L(s')$, and hence we can use Lemma 7.4 to obtain $\mathtt{fts}^+(fks) \models s \overset{f_\perp}{\leftrightarrows}_f s'$. Combining these results gives us $\mathtt{fts}^+(fks) \models s \overset{\varphi \vee f_\perp}{\leftrightarrows}_f s'$.

Suppose $\mathtt{fts}^+(fks) \models s \overset{\varphi \vee f_\perp}{\leftrightarrows}_f s'$. Hence we also have $\mathtt{fts}^+(fks) \models s \overset{\varphi}{\leftrightarrows}_f s'$. By Theorem 4.5 this implies $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathtt{fts}^+(fks)|_P \models s \leftrightarrows s'$, which is equivalent to $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathtt{lts}(fks|_P) \models s \leftrightarrows s'$ by Lemma 7.3. Using Theorem 7.1 we obtain $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow fks|_P \models s \leftrightarrows s'$. Since $\mathtt{fts}^+(fks) \models s \overset{f_\perp}{\leftrightarrows}_f s'$, we use Lemma 7.4 to obtain $L(s) = L(s')$. Using this we can conclude that $fks \models s \overset{\varphi}{\leftrightarrows}_f s'$, by Theorem 4.2.

□

Theorem 7.8. *Let fts be an arbitrary reversible FTS. Then the FTS each fts′* $\in \leftrightarrows_{cf+}\text{-min}_{\mathtt{FTS}}(fts)$ *is also reversible.*

Proof. Let $fts = (S, \mathcal{A}, \theta, s_*, \mathcal{F}, \mathcal{P})$ be a reversible FTS. Let $fts' = (S', \mathcal{A}, \theta', s'_*, \mathcal{F}, \mathcal{P})$ be a coherent$^+$ feature bisimulation quotient of *fts*. Let ϱ and ϱ' denote the reachability functions of *fts* and *fts′*, respectively. We show that *fts′* is reversible.

1. $\mathcal{A} = 2^{AP} \cup \{\perp\}$ for some set AP follows directly from reversibility of *fts*.
2. $f_\perp \in \mathcal{F}$ and $\{f_\perp\} \in \mathcal{P}$ and $\forall P \in \mathcal{P}\colon (\, P \neq \{f_\perp\} \Rightarrow P \not\models f_\perp \,)$ follows directly from reversibility of *fts*.
3. Since *fts* is reversible, we know that $S = (S_\perp, \bar{S})$. We furthermore know that $s_\perp \xrightarrow{\perp|\psi}$ for some $\psi \in \mathbb{B}(\mathcal{F})$ such that $f_\perp \Rightarrow \psi$, for all $s_\perp \in S_\perp$, and that $\bar{s} \overset{\perp}{\not\rightarrow}$ for all $\bar{s} \in \bar{S}$. By definition of feature bisimulation we derive that $s \overset{f_\perp}{\leftrightarrows}_f t$ implies that $s, t \in S_\perp$ or $s, t \in \bar{S}$, for all $s, t \in S$. By definition of $\leftrightarrows_{cf+}\text{-min}_{\mathtt{FTS}}$ it follows that $s, t \in C$ implies $s, t \in S_\perp$ or $s, t \in \bar{S}$, for all $C \in S'$. Hence we can find a partition $(S'_\perp, \bar{S}')$ of S' where $S'_\perp = \{C \in S' \mid \forall C \subseteq S_\perp\}$ and $\bar{S}' = \{C \in S' \mid \forall C \subseteq \bar{S}\}$.
4. Since $s_* \in S_\perp$ and $s_* \in s'_*$, it follows that $s'_* \in S'_\perp$.
5. For all $C_1, C_2 \in S'_\perp$, $\bar{C}_1, \bar{C}_2 \in \bar{S}'$, and $\alpha, \alpha' \in \mathcal{A}_\tau$ we have
   (a) For all $s \in C_1$ and $t \in C_2$ we have $s, t \in S_\perp$ and hence $(f_\perp \wedge \theta(s, \alpha, t)) \sim_{\mathcal{P}} \mathsf{false}$. By definition of $\leftrightarrows_{cf+}\text{-min}_{\mathtt{FTS}}$ it follows that $(f_\perp \wedge \theta'(C_1, \alpha, C_2)) \sim_{\mathcal{P}} \mathsf{false}$.
   (b) For all $s \in C_1$ and $t \in \bar{C}_1$ we have $s \in S_\perp$ and $t \in \bar{S}$ and hence $\theta(s, \perp, t) \not\sim_{\mathcal{P}} \mathsf{false}$ implies $f_\perp \Rightarrow_{\mathcal{P}} \theta(s, \perp, t)$. By definition of $\leftrightarrows_{cf+}\text{-min}_{\mathtt{FTS}}$ it follows that $\theta'(C_1, \perp, \bar{C}_1) \not\sim_{\mathcal{P}} \mathsf{false}$ implies $f_\perp \Rightarrow_{\mathcal{P}} \theta'(C_1, \perp, \bar{C}_1)$.
   (c) For all $s \in C_1$ we have $s \in S_\perp$ and hence $(f_\perp \vee \varrho(s)) \Rightarrow_{\mathcal{P}} \bigvee_{\bar{s} \in \bar{S}} \theta(s, \perp, \bar{s})$. We have $\varrho'(C_1) \sim_{\mathcal{P}} \bigvee_{s \in C_1} \varrho(s)$, and hence we can derive

$$f_\perp \vee \varrho'(C_1) \Rightarrow_{\mathcal{P}} \bigvee_{s \in C_1} \bigvee_{\bar{s} \in \bar{S}} \theta(s, \perp, \bar{s}).$$

By definition of $\leftrightarrows_{cf+}$-min$_{\mathtt{FTS}}$ we have

$$\bigvee_{s\in C_1}\bigvee_{\bar{s}\in\bar{S}}\theta(s,\perp,\bar{s}) \sim_{\mathcal{P}} \bigvee_{\bar{C}\in\bar{S}'}\theta'(C_1,\perp,\bar{C}),$$

and hence we have $f_\perp \vee \varrho'(C_1) \Rightarrow_{\mathcal{P}} \bigvee_{\bar{C}\in\bar{S}'}\theta'(C_1,\perp,\bar{C})$.

(d) For all $s\in\bar{C}_1$ and $t\in\bar{C}_2$ we have $s,t\in\bar{S}$ and hence $\theta(s,\alpha,t)\sim_{\mathcal{P}}$ false. By definition of $\leftrightarrows_{cf+}$-min$_{\mathtt{FTS}}$ it follows that $\theta'(\bar{C}_1,\alpha,\bar{C}_2)\sim_{\mathcal{P}}$ false.

(e) Assume $\theta'(\bar{C}_1,\alpha,C) \not\sim_{\mathcal{P}}$ false and $\theta'(\bar{C}_1,\alpha',C') \not\sim_{\mathcal{P}}$ false, for some $C,C'\in S'_\perp$. By definition of $\leftrightarrows_{cf+}$-min$_{\mathtt{FTS}}$ this implies there exist $s,t\in\bar{C}_1$ such that $\theta(s,\alpha,s') \not\sim_{\mathcal{P}}$ false and $\theta(t,\alpha',t') \not\sim_{\mathcal{P}}$ false, for some $s'\in C$ and some $t'\in C'$, and hence $s',t'\in S_\perp$. By reversibility of *fts*, using condition 6(a) it follows that $\alpha = L(s')$, from which we derive $f_\perp \Rightarrow_{\mathcal{P}} \theta(s,\alpha,s')$ using conditions 6(b) and 5(b). By definition of $\leftrightarrows_{cf+}$-min$_{\mathtt{FTS}}$ we know that $fts \models s \overset{f_\perp}{\longleftrightarrow}_f t$, and hence t must be able to mimic the α-step for $f_\perp$. By conditions 5(d) and 5(e) we know that for all $\beta,\beta'\in\mathcal{A}_\tau$ it holds that $t\xrightarrow{\beta}$ and $t\xrightarrow{\beta'}$ implies $\beta=\beta'$, from which it follows that the transfer conditions can only be satisfied if $\alpha=\alpha'$, which satisfies this condition.

6. From conditions 5(c) and 6(b) it follows that each state $s_\perp\in S_\perp$ has at least one incoming transition with label $L(s_\perp)$.

   We show that for all $s,t\in C_\perp$, for each $C_\perp\in S'_\perp$, we have $L(s)=L(t)$. Using conditions 5(b), 5(c) and 6(b) we derive that $\theta(s,\perp,\bar{s}) \sim_{\mathcal{P}} \theta(\bar{s},L(s),s)$ and $f_\perp \Rightarrow_{\mathcal{P}} \theta(s,\perp,\bar{s})$, for some $\bar{s}\in\bar{S}$. By definition of $\leftrightarrows_{cf+}$-min$_{\mathtt{FTS}}$ we know that $fts \models s \overset{f_\perp}{\longleftrightarrow}_f t$ and hence $f_\perp \Rightarrow_{\mathcal{P}} \theta(t,\perp,\bar{t})$, for some $\bar{t}\in\bar{S}$ such that $fts\models\bar{s}\overset{f_\perp}{\longleftrightarrow}_f\bar{t}$. By conditions 5($d$) and 6($a$) it follows that $\theta(\bar{t},\alpha,t') \not\sim_{\mathcal{P}}$ false implies $\alpha=L(t)$, for all $t'\in S$, and hence it must be the case that $L(s)=L(t)$ for the transfer conditions to be satisfied.

   Now we can define $L(C_\perp)=L(s)$, for all $s\in C_\perp$, for all $C_\perp\in S'_\perp$.

   For all $C_\perp,C'_\perp\in S'_\perp$, $\bar{C}\in\bar{S}'$, and $\alpha\in\mathcal{A}_\tau$ we have:

   (a) For all $s\in\bar{C}$ we have $s\in\bar{S}$ and hence $\theta(s,\alpha,t) \not\sim_{\mathcal{P}}$ false implies $\alpha=L(t)$, for all $t\in C_\perp$. By definition of $\leftrightarrows_{cf+}$-min$_{\mathtt{FTS}}$, and since $L(C_\perp)=L(t)$, for all $t\in C_\perp$, it follows that $\theta'(\bar{C},\alpha,C_\perp) \not\sim_{\mathcal{P}}$ false implies $\alpha=L(C_\perp)$.

   (b) For each pair of states $(s,t)\in\bar{C}\times C_\perp$ we have $\theta(s,L(t),t)\sim_{\mathcal{P}}\theta(t,\perp,s)$. Hence, by definition of $\leftrightarrows_{cf+}$-min$_{\mathtt{FTS}}$ it follows that

   $$\theta'(\bar{C},L(C_\perp),C_\perp) = \bigvee_{(s,t)\in\bar{C}\times C_\perp}\varrho(s)\wedge\theta(s,L(t),t) \sim_{\mathcal{P}} \bigvee_{(s,t)\in\bar{C}\times C_\perp}\varrho(t)\wedge\theta(t,\perp,s) = \theta'(C_\perp,\perp,\bar{C}).$$

   (c) For each pair of states $(t,s)\in C'_\perp\times C_\perp$ we have $L(t)=L(s)$ and $\theta(t,\alpha,s) \not\sim_{\mathcal{P}}$ false implies $\alpha=\tau$. By definition of $\leftrightarrows_{cf+}$-min$_{\mathtt{FTS}}$ it follows that $L(C'_\perp)=L(C_\perp)$ and $\theta'(C'_\perp,\alpha,C_\perp) \not\sim_{\mathcal{P}}$ false implies $\alpha=\tau$.

   (d) For each pair of states $(t,s)\in C'_\perp\times C_\perp$ we have $L(t)\neq L(s)$ and $\theta(t,\alpha,s) \not\sim_{\mathcal{P}}$ false implies $\alpha=L(s)$. By definition of $\leftrightarrows_{cf+}$-min$_{\mathtt{FTS}}$ it follows that $L(C'_\perp)\neq L(C_\perp)$ and $\theta'(C'_\perp,\alpha,C_\perp) \not\sim_{\mathcal{P}}$ false implies $\alpha=L(s)=L(C_\perp)$.

We conclude that fts' is indeed reversible. □

Lemma 7.7. *Let fks be an FKS. For each FTS* $fts \in \leftrightarrows_{cf+}\text{-min}_{\texttt{FTS}}(\texttt{fts}^{+}(fks))$ *there exists an FKS* $fks' \in \leftrightarrows_{cf}\text{-min}_{\texttt{FKS}}(fks)$ *such that* $fks' = \texttt{fts}^{+\,-1}(fts)$.

Proof. Let $fks = (S_1, AP, \theta_1, L, s_{*\,1}, \mathcal{F}_1, \mathcal{P}_1)$ be an FKS with reachability function ϱ_1. Let $\texttt{fts}^{+}(fks) = (S_2, \mathcal{A}, \theta_2, s_{*\,2}, \mathcal{F}_2, \mathcal{P}_2)$ be its corresponding FTS with reachability function ϱ_2, and let $fts' = (S'_2, \mathcal{A}, \theta'_2, s'_{*\,2}, \mathcal{F}_2, \mathcal{P}_2)$ be a coherent^{+} quotient of $\texttt{fts}^{+}(fks)$. Lastly, let $\texttt{fts}^{+\,-1}(fts') = (S_3, AP', \theta_3, L', s_{*\,3}, \mathcal{F}_3, \mathcal{P}_3)$ denote the corresponding FKS of fts'.

By definition of $\leftrightarrows_{cf+}\text{-min}_{\texttt{FTS}}$, we know that S'_2 is a partition of S_2. Furthermore we know that for each class $C \in S'_2$ we have $C \subseteq S_1$ or $C \cap S_1 = \emptyset$. For each class $C \in S'_2$ such that $C \subseteq S_1$, for each pair of states $s, t \in C$ we have that $\texttt{fts}^{+}(fks) \models s \leftrightarrows_{cf+} t$, and hence $fks \models s \leftrightarrows_{cf} t$, by Theorem 7.3.

From this it follows that $S'_1 = \{C \in S'_2 \mid C \subseteq S_1\}$ is the statespace of a coherent quotient of fks. Let $fks' = (S'_1, AP, \theta'_1, L'', s'_{*\,1}, \mathcal{F}_1, \mathcal{P}_1)$ denote this quotient. We show that $fks' = \texttt{fts}^{+\,-1}(fts')$.

We establish the isomorphism by proving that there are isomorphism between S'_1 and S_3, AP and AP', θ'_1 and θ_3, L' and L'', $s'_{*\,1}$ and $s_{*\,3}$, $\mathcal{F}_1$ and $\mathcal{F}_3$, and $\mathcal{P}_1$ and $\mathcal{P}_3$.

- We find that the set of states of $\texttt{fts}^{+\,-1}(fts')$ is

  $$S_3 = \{C \in S'_2 \mid C \xrightarrow{\perp}\}$$

  By definition of $\texttt{fts}^{+}$ and $\leftrightarrows_{cf}\text{-min}_{\texttt{FKS}}$ we know that this is equal to $\{\, C \in S'_2 \mid C \subseteq S_1 \,\} = S'_1$.

- The set of actions is unchanged by $\leftrightarrows_{cf}\text{-min}_{\texttt{FKS}}$, and hence $AP = AP'$ follows from $\texttt{fts}^{+\,-1} \circ \texttt{fts}^{+} = \texttt{Id}$.

- By definition of $\leftrightarrows_{cf}\text{-min}_{\texttt{FKS}}$ we have that $\theta'_1 \colon S'_1 \times S'_1 \to \mathbb{B}(\mathcal{F})$ is defined such that, for all $C_1, C_2 \in S'_1$:

  $$\theta'_1(C_1, C_2) = \bigvee\{\, \theta_1(s,t) \wedge \varrho_1(s) \mid s \in C_1 \wedge t \in C_2 \,\}.$$

  By definition of $\texttt{fts}^{+}$ we have that $\theta_2 \colon S_2 \times \mathcal{A}_\tau \times S_2 \to \mathbb{B}(\mathcal{F})$ is defined such that, for all $s, t \in S_2$ and $\alpha \in \mathcal{A}_\tau$:

  $$\theta_2(s,\alpha,t) = \begin{cases} \mathsf{true} & \text{if } s \in S_1 \wedge t = \bar{s} \wedge \alpha = \perp \\ \mathsf{true} & \text{if } t \in S_2 \wedge s = \bar{t} \wedge \alpha = L(t) \\ \theta(s,t) \wedge \neg f_\perp & \text{if } s, t \in S_1 \wedge L(s) = L(t) \wedge \alpha = \tau \\ \theta(s,t) \wedge \neg f_\perp & \text{if } s, t \in S_1 \wedge L(s) \neq L(t) \wedge \alpha = L(t) \\ \mathsf{false} & \text{otherwise} \end{cases}$$

  By definition of $\leftrightarrows_{cf+}\text{-min}_{\texttt{FTS}}$ we have that $\theta'_2 \colon S'_2 \times \mathcal{A}_\tau \times S'_2 \to \mathbb{B}(\mathcal{F})$ is defined such that, for all $C_1, C_2 \in S'_2$ and $\alpha \in \mathcal{A}_\tau$:

  $$\theta'_2(C_1, \alpha, C_2) = \bigvee\{\, \theta_2(s,\alpha,t) \wedge (\varrho_2(s) \vee f_\perp) \mid s \in C_1 \wedge t \in C_2 \,\},$$

which is equal to:

$$\theta'_2(C_1,\alpha,C_2) = \begin{cases} \bigvee\{\varrho_2(s) \vee f_\perp \mid s \in C_1 \wedge \bar{s} \in C_2\} & \text{if } C_1 \in S'_1 \wedge \alpha = \perp \\ \bigvee\{\varrho_2(t) \vee f_\perp \mid t \in C_2 \wedge \bar{t} \in C_1\} & \text{if } C_2 \in S'_1 \wedge \alpha = L''(C_2) \\ \bigvee\{\theta(s,t) \wedge \neg f_\perp \wedge \varrho_2(s) \mid s,t \in C_1,C_2\} & \text{if } C_1,C_2 \in S'_1 \wedge L''(C_1) = L''(C_2) \wedge \alpha = \tau \\ \qquad\qquad \mid s,t \in C_1,C_2\} & \\ \bigvee\{\theta(s,t) \wedge \neg f_\perp \wedge \varrho_2(s) \mid s,t \in C_1,C_2\} & \text{if } C_1,C_2 \in S'_1 \wedge L''(C_1) \neq L''(C_2) \wedge \alpha = L''(C_2 \\ \mid s,t \in C_1,C_2\} & \\ \mathsf{false} & \text{otherwise} \end{cases}$$

By definition of $\mathtt{fts}^{+-1}$ we have that $\theta_3 \colon S_3 \times S_3 \to \mathbb{B}(\mathcal{F})$ is defined such that, for all $C_1, C_2 \in S_3$:

$$\theta_3(C_1,C_2) = \theta'_2(C_1,L'(C_2),C_2) \vee \theta'_2(C_1,\tau,C_2).$$

Using that $L' = L''$, this is equal to:

$$\theta_3(C_1,C_2) = \bigvee\{\theta(s,t) \wedge \neg f_\perp \wedge \varrho_2(s) \mid s,t \in C_1,C_2\}.$$

Using that $S'_1 = S_3$ and that $\varrho_2(s) \wedge \neg f_\perp \sim_{\mathcal{P}} \varrho_1(s)$, for all $s \in S_1$, we conclude that $\theta'_1 = \theta_3$.

- For all $C_2 \in S_3$, $L'(C_2) = a$ for the unique $a \in 2^{AP}$ such that $C_1 \xrightarrow{a} C_2$ in fts', for some $C_1 \in S'_2$. Pick some $s \in C_2$. By construction of θ'_2 we know we can pick C_1 such that $\bar{s} \in C_1$ and $a = L''(C_2)$. Hence $L' = L''$.
- The initial state is unchanged by the embedding $\mathtt{fts}^{+-1}$, and hence we find $s_{*3} = s'_{*2}$, which is the class $C \in S'_2$ such that $s_{*2} \in C$. Since $s_{*2} = s_{*1}$ and $s_{*1} \in s'_{*1}$, we derive that $s_{*3} = s'_{*1}$.
- The set of features is unchanged by $\leftrightarrows_{cf}\text{-min}_{\mathtt{FKS}}$, and hence $\mathcal{F}_1 = \mathcal{F}_3$ follows from $\mathtt{fts}^{+-1} \circ \mathtt{fts}^{+} = \mathtt{Id}$.
- The set of products is unchanged by $\leftrightarrows_{cf}\text{-min}_{\mathtt{FKS}}$, and hence $\mathcal{P}_1 = \mathcal{P}_3$ follows from $\mathtt{fts}^{+-1} \circ \mathtt{fts}^{+} = \mathtt{Id}$.

□

Lemma 7.8. *Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS. Then, for all $P \in \mathcal{P}$ and for all $s,t \in s$ we have $\mathtt{fks}(fts)|_P \models s \leftrightarrows t \Leftrightarrow \mathtt{ks}(fts|_P) \models s \leftrightarrows t$.*

Proof. Let $fts = (S, \mathcal{A}, \theta, s_*)$ be an FTS, and let $P \in \mathcal{P}$ be product. Let $\mathtt{fks}(fts)|_P = (S_1, AP, \to_1, s_*)$ and $\mathtt{ks}(fts|_P) = (S_2, AP, \to_2, s_*)$. Note that $S_2 \subseteq S_1$, such that all states in $S_1 \setminus S_2$ are unreachable in $\mathtt{fks}(fts)|_P$. Furthermore $\to_2 \subseteq \to_1$, such that $\to_1 \setminus \to_2 \subseteq (S_1 \setminus S_2) \times S_2$. Let $s,t \in S$ be two states.

Assume $\mathtt{fks}(fts)|_P \models s \leftrightarrows t$, and let $R \subset S_1 \times S_1$ be the smallest bisimulation relation witnessing this. Since R is the smallest relation all states in $S_1 \setminus S_2$ are unreachable in $\mathtt{fks}(fts)|_P$, it follows that $R \subset S_2 \times S_2$. Since $\to_2 \subseteq \to_1$ we conclude that R is also a bisimulation relation for $\mathtt{ks}(fts|_P)$.

Assume $\mathtt{ks}(\mathit{fts}|_P) \models s \leftrightarrows t$, and let $R \subset S_2 \times S_2$ be a bisimulation relation witnessing this. Since $S_1 \setminus S_2$, it follows that $R \subset S_1 \times S_1$. Since $\rightarrow_1 \setminus \rightarrow_2 \subseteq (S_1 \setminus S_2) \times S_2$ we conclude that R is also a bisimulation relation for $\mathtt{fks}(\mathit{fts})|_P$. □

Theorem 7.11. *Let $\mathit{fts} = (S, \mathcal{A}, \theta, s_*)$ be an FTS.Then, for all $s, s' \in S$, and $\varphi \in \mathbb{B}(\mathcal{F})$, we have $\mathit{fts} \models s \overset{\varphi}{\leftrightarrows}_f s'$ if and only if $\mathtt{fks}(\mathit{fts}) \models s \overset{\varphi}{\leftrightarrows}_f s'$.*

Proof. Let $\mathit{fts} = (S, \mathcal{A}, \theta, s_*)$ be an FTS, and let $s, s' \in S$ and $\varphi \in \mathbb{B}(\mathcal{F})$.

Suppose $\mathit{fts} \models s \overset{\varphi}{\leftrightarrows}_f s'$. By Theorem 4.5 this implies $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathit{fts}|_P \models s \leftrightarrows s'$. By Theorem 7.10 we get $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathtt{ks}(\mathit{fts}|_P) \models s \leftrightarrows s'$. Using Lemma 7.8 we get $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathtt{fks}(\mathit{fts})|_P \models s \leftrightarrows s'$. By definition of the embedding $\mathtt{fks}(\mathit{fks})$, we know that the labels of both s and s' in $\mathtt{fks}(\mathit{fts})$ are $\{\bot\}$. Hence we can use Theorem 4.2 to obtain $\mathtt{fks}(\mathit{fts}) \models s \overset{\varphi}{\leftrightarrows}_f s'$.

Suppose $\mathtt{fks}(\mathit{fts}) \models s \overset{\varphi}{\leftrightarrows}_f s'$. By Theorem 4.1 this implies $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathtt{fks}(\mathit{fts})|_P \models s \leftrightarrows s'$. By Lemma 7.8 we get $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathtt{ks}(\mathit{fts}|_P) \models s \leftrightarrows s'$. Using Theorem 7.10 we obtain $\forall P \in \mathcal{P}\colon P \models \varphi \Rightarrow \mathit{fts}|_P \models s \leftrightarrows s'$, from which we conclude that $\mathit{fts} \models s \overset{\varphi}{\leftrightarrows}_f s'$, by Theorem 4.6. □

Lemma 7.10. *Let the FKS $\mathit{fks} = (S, AP, \theta, L, s_*)$ be a reversible FKS. For states $s, t \in S$ such that $L(s) \neq \{\bot\}$ and $L(t) \neq \{\bot\}$, and states $s', t' \in S$ such that $s \rightarrow s'$ and $t \rightarrow t'$ we have, for all $\varphi \in \mathbb{B}(\mathcal{F})$:*

$$\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t \;\Leftrightarrow\; (\,\mathit{fks} \models s' \overset{\varphi}{\leftrightarrows}_f t' \wedge L(s) = L(t)\,).$$

Proof. Let $\mathit{fks} = (S, AP, \theta, L, s_*)$ be a reversible FKS. Let $s, t \in S$ be such that $L(s) \neq \{\bot\}$ and $L(t) \neq \{\bot\}$. Let states s' and t' be targets of s and t, respectively.

Assume $\mathit{fks} \models s \overset{\varphi}{\leftrightarrows}_f t$, for some $\varphi \in \mathbb{B}(\mathcal{F})$. By definition of feature bisimilarity for FKS it immediately follows that $L(s) = L(t)$. By definition of reversibility both s and t have only one outgoing transition: $s \xrightarrow{\mathsf{true}} s'$ and $t \xrightarrow{\mathsf{true}} t'$. Hence, in order to satisfy the transfer condition if $\varphi \not\sim_{\mathcal{P}} \mathsf{false}$ it must be the case that $\mathit{fks} \models s' \overset{\varphi}{\leftrightarrows}_f t'$. If $\varphi \sim_{\mathcal{P}} \mathsf{false}$ it trivially follows that $\mathit{fks} \models s' \overset{\varphi}{\leftrightarrows}_f t'$ as long as $L(s') = L(t')$. By definition of reversibility we find that $L(s') = \{\bot\} = L(t')$, and hence this case is also satisfied.

Assume $\mathit{fks} \models s' \overset{\varphi}{\leftrightarrows}_f t'$, let $R \subset S \times \widehat{\mathbb{B}}(\mathcal{F}) \times S$ be the relation that witnesses this, and $L(s) = L(t)$. We show that $R \cup (s, \hat{\varphi}, t)$ is a feature bisimulation relation for fks as well. By assumption we have $L(s) = L(t)$. By definition of reversibility both s and t have only one outgoing transition: $s \xrightarrow{\mathsf{true}} s'$ and $t \xrightarrow{\mathsf{true}} t'$. Since the tuple $(s', \hat{\varphi}, t')$ is included in R, the transfer condition for the tuple $(s, \hat{\varphi}, t)$ is immediately satisfied. □

Appendix F

FTS Toolset User Manual

The FTS toolset was developed to define, abstract and reduce SPLs. The SPLs are defined using FDs and FTSs, and a conversion tool was created to convert SPL specifications to mCRL2 specifications, which allows for automated model checking using the mCRL2 tool set.

The different file formats used in the toolset are described in Section F.1. The different tools and their usage are discussed in Section F.2.

F.1 File formats

This section describes the purpose and syntax of the eight different file formats of the FTS Toolset. An example of each type of file will also be provided.

F.1.1 SPL (.xml)

An SPL specification defines a software product line.

Syntax

The syntax of an SPL file consists of an XML schema. An overview of the schema is provided at page 152. Optional XML elements are displayed in square brackets. The schema has an `spl` root element, with a mandatory `name` attribute. This attribute specifies the name of the SPL. The `spl` root element has a number of child elements:

- `feature_diagram` is an element that specifies the products of this SPL. It has the following child elements:
  - `feature_model` is an element specifying the FD file (Section F.1.2) of this SPL. It has a mandatory `name` attribute specifying the path from this SPL file to the FD file.
  - `products` is an optional element specifying the Products file (Section F.1.3) of this SPL. It has a mandatory `name` attribute specifying the path from this SPL file to the Products file.
  - `expressions` is an optional element specifying the Expressions file (Section F.1.4) of this SPL. It has a mandatory `name` attribute specifying the path from this SPL file to the Expressions file.
- `feature_transitions_systems` is an element that specifies the behavior of this SPL, as a number of FTS running in parallel. It has at least one `fts` child element with mandatory `name` attribute specifying the path from this SPL file to its corresponding FTS file (Section F.1.5).
- `datatypes` is an optional element that specifies user-defined data-types. Each data-type is specified using a `struct` element that has a mandatory `name` attribute specifying the name of the data-type. Each `struct` element has at least on `value` child element with a mandatory `name` attribute, specifying the value name. Note that all values should be unique.
- `parameterized_actions` is an optional element that specifies actions with parameters. Each parameterized action is specified using an `action` element that has a mandatory `name` attribute specifying the name of the action. Its parameters are defined using `parameter` child elements, each having a mandatory attributes `id`, specifying whether this parameter is the first parameter of this actions, the second parameter of this action, etc. Furthermore each parameter has a mandatory `type` attribute specifying the type of the parameter. The value of this parameter can be the name of any user-defined type in the `datatypes` section, or one of the predefined mCRL2 sorts[1] (`Bool`, `Pos`, `Nat`, `Int` or `Real`).

[1] `http://www.mcrl2.org/release/user_manual/language_reference/data.html#predefined-sorts`

- `communication` is an optional that specifies which multi-actions are renamed to a single action. Each multi-action is specified using a `multiaction` element that has a mandatory `name` attributed specifying the name of the renamed multi-action. The separate actions this multi-actions consists of are specified using `action` child elements, each having a mandatory `name attribute`. All of the actions of a single multi-action must have the same parameter types. Each multi-action element has an optional `type` attribute that should be assigned the value `optional` if communication is not enforced (i.e. the actions of the multi-action can also be executed individually).

Example

A small example SPL file specifying the vending machine product line is provided below.

```
<spl name="VendingMachine_SPL">
<feature_diagram>
    <feature_model name="VM_FD.xml" />
</feature_diagram>
<feature_transition_systems>
    <fts name="VM_FTS.fts" />
</feature_transition_systems>
<datatypes>
    <struct name="Beverage">
        <value name="Coffee" />
        <value name="Tea" />
        <value name="Water" />
    </struct>
</datatypes>
<parameterized_actions>
    <action name="order">
        <parameter id="1" type="Beverage" />
    </action>
    <action name="serve">
        <parameter id="1" type="Beverage" />
    </action>
</parameterized_actions>
</spl>
```

The SPL file XML schema:

```
<spl name="SPL_name">

    <feature_diagram>
        <feature_model name="FD_file" />
        [ <products name="Products_file" /> ]
        [ <expressions name="Expressions_file" /> ]
    </feature_diagram>

    <feature_transition_systems>
        <fts name="FTS_file" />
        [ <fts name="FTS_file2" /> ]
        ...
    </feature_transition_systems>

    [ <datatypes>
        <struct name="Struct1">
            <value name="Value1" />
            [ <value name="Value2" /> ]
            ...
        </struct>
        ...
    </datatypes> ]

    [ <parameterized_actions>
        <action name="Action1">
            <parameter id="1" type="Struct1" />
            [ <parameter id="2" type="Int" /> ]
            ...
        </action>
        ...
    </parameterized_actions> ]

    [ <communication>
        <multiaction name="MultiAction1" [type="optional"]>
            <action name="Act1">
            <action name="Act2">
            [ <action name="Act3"> ]
            ...
        </multiaction>
        ...
    </communication> ]
```

```
</spl>
```

F.1.2 FD (.xml)

An FD specification defines an attributed feature model defining the products of a software product line.

Syntax

The syntax of an FD file consists of an XML schema. An overview of the schema is provided at page 156. Optional XML elements are displayed in square brackets. The schema has an `feature_model` root element, with a mandatory `name` attribute. This attribute specifies the name of the SPL. The `spl` root element has a number of child elements:

- `attributes` is an element that specifies that attributes of the FD. Each attribute is specified using a `attribute` element that has a mandatory `name` attribute specifying the name of the attribute. It furthermore has an optional `type` attribute, for which the only supported value is `Integer`. If the `type` attribute is not used, this attribute is assigned the type `Integer`. Lastly it has an optional `default` attribute, specifying the default value this attribute is assigned for each feature. If the `default` attribute is not used, default value `0` is used.
- `feature_tree` is an element that specifies the feature of th FD and their tree-structure. It has a single child `feature` element with mandatory `name`, `id` and `type` attributes, specifying the name, short identifier and type of the feature, respectively. The value of the `type` attribute must be `'root'`. The root `feature` element has an arbitrary number of `feature_tree_element` child elements. A `feature_tree_element` is either a `feature_element` or a `group_element`, where:
  - a `feature_element` is a `feature` element. It has mandatory `name` and *id* attributes, specifying the name and short identifier of the feature. Furthermore it has an optional `type` attribute that should be assigned the value `mandatory` if the feature has an 'mandatory' link to its parent feature, meaning that when a product contains the parent feature, this feature must also be included in the product. If the value `optional` is assigned to the attribute `type`, or the `type` attribute is not used at all, the feature has a 'optional' link to its parent feature, meaning that when a product contains the parent feature, this feature may also be in included in the product, but does not have to be.

    The `feature` element may have `attribute` child elements to assign values to the attributes declared in the `attributes` section for this feature. An `attribute` element has mandatory `value` and `name` attributes, specifying a value and the name of the attribute to assign this value to, respectively. Lastly, the `feature` element may have an arbitrary amount of child `feature_tree_element` elements, specifying the child features of this feature.

  - a `group_element` is a `group` element. It has a mandatory `type` attribute that should be assigned either the value `XOR` or the value `OR`. If the type is `XOR` it means that when a product contains the parent feature of this `group` element, exactly one of the features contained in the `group` element must be in the product. Likewise, if the type is `OR` it means that when a product contains the parent feature of this `group` element, at least one of the features contained in the `group` element must be in the product. A `group` element must have at least on child `feature_element` element, specifying the features included in this group.

- `constraints` is an optional element that specifies additional constraints on the FD. Each constraint is specified using a `constraint` element that has a mandatory `type` attribute specifying the type of the constraint. The supported values for the `type` attribute are `CTC` and `attribute`. If the value `CTC` is used, the `constraint` element must have `value` attribute specifying a cross-tree constraint over the short feature identifiers (`id` attribute of `feature` elements) in disjunctive normal formal form, using `|` for disjunction, `&` for conjunction and `!` for negation. If the value `attribute` is used for the `type` attribute, the `constraint` element must have `name`, `operator` and `value` attributes, specifying a constraint on the sum over the attribute values for each product, for the attribute specified by `name`. Supported values for the `operator` attribute are `<` (less than), `<=` (at most), `==` (equal), `>=` (at least), `>` (greater than) and `!=` (not equal). The value of the `value` attribute should be an integer.

Example

A small example FD file specifying the products of the vending machine product line is provided below.

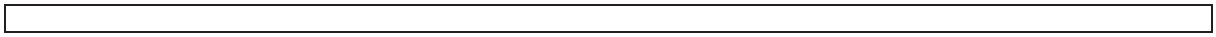
```
<feature_model name="VendingMachine_FD">
<attributes/>
<feature_tree>
    <feature name='Machine' id='m' type='root'>
        <feature name='Beverage' id='b' type='mandatory'>
            <feature name='Coffee' id='c' type='mandatory'/>
            <feature name='Tea' id='t' type='optional'/>
            <feature name='Water' id='w' type='optional'/>
        </feature>
        <feature name='FreeDrinks' id='f' type='optional'/>
        <feature name='CancelPurchase' id='x' type='optional'/>
    </feature>
</feature_tree>
<constraints>
    <constraint type='CTC' value='!w|f'/>
    <constraint type='CTC' value='!f|!x'/>
</constraints>
</feature_model>
```

The FD file XML schema:

```
<feature_model name="FD_name">

    <attributes>
        [ <attribute name="Attribute1" [type="Integer"]
                     default="DefaultValue"] /> ]
        ...
    </attributes>

    <feature_tree>
        <feature name="RootFeature" id="R" type="root">
            [ <feature_tree_element /> ]
            ...
        </feature>
    </feature_tree>

    <constraints>
        [ <constraint type="CTC" value="ConstraintValue"> ]
        ...
        [ <constraint type="attribute" name="Attribute1"
                      operator="&lt;=" value="ConstraintValue" /> ]
        ...
    </constraints>

</feature_model>
```

The XML schema of the `feature_tree_element` element:

```
<feature_tree_element /> ::= <feature_element/> | <group_element/>

<feature_element/> ::=
<feature name="Feature1" id="F1" [type="mandatory"]>
    [ <attribute name="Attribute1" value="AttributeValue" /> ]
    ...
    [ <feature_tree_element /> ]
    ...
</feature>

<group_element/> ::=
<group type="XOR">
    <feature_element />
    [ <feature_element /> ]
    ...
</group>
```



F.1.3 Products (.prod)

A Products file explicitly contains the products of a software product line.

Syntax

The syntax of an Products file consists of a number of lines. The first line consists of a single number (`n`) representing the number of products in the SPL. The following `n` lines specify the `n` different products, and have the following syntax:

```
product ::= id ‘ ’ product

id ::= number
product ::= ‘[‘ feature_list ‘]’

feature_list ::= feature_identifier | feature_list ‘, ’ feature_list
feature_identifier ::= string
```

Here:

- `id` is a natural number less than `n`. Each of the `n` product lines must have a unique id.
- `product` is a comma separated list of feature-identifiers, representing the features that form this product.

Example

A small example Products file explicitly specifying the products of the vending machine product line is provided below.

```
8
0 [m, w, f, t, c, b]
1 [w, f, m, c, b]
2 [m, t, c, b, x]
3 [f, m, t, c, b]
4 [m, c, b, x]
5 [f, m, c, b]
6 [m, t, c, b]
7 [m, c, b]
```

F.1.4 Expressions (.expr)

An Expressions file contains the expression values corresponding to the feature-identifiers of the software product line.

Syntax

The syntax of an Products file consists of a number of lines. The first line consists of a single feature-identifier representing the root feature of the SPL. The second line consists of a space separated list of feature-identifiers representing the core features of the SPL, i.e. the features that are present in all products of the SPL. The third line consists of a space-separated list of feature-identifiers representing the remaining (non-core) features of the SPL (`n` in total). The following `n` lines specify expression values for the non-core features, and have the following syntax

```
expression ::= feature_identifier ‘ ’ value

feature_identifier ::= string
expression_value ::= number
```

Here:

- `feature_identifier` is the feature-identifier of one of the non-core features. Each of the `n` expression lines must have a unique feature-identifier.
- `value` is a natural number less than 2^p, where `p` is the number of products of the SPL. Here, the bit at position `x` of the `p`-bit binary representation of `value` is 1 iff product `x` (as specified in the corresponding Products file (Section F.1.3)) contains `feature_identifier`, where position 0 corresponds to the right-most bit, and position `p-1` corresponds to the left-most bit.

Example

A small example Expressions file specifying the expression values corresponding to the feature-identifiers of the vending machine product line is provided below.

```
m
m b c
t w f x
t 77
w 3
f 43
x 20
```

F.1.5 FTS (.fts)

An FTS specification defines a feature labeled transition system.

Syntax

The syntax of an FTS file consists of a number of lines. The first line consists of a single number representing the number of states in the FTS. The second line contains the set `actions` of action names separated by spaces. The third line contains the set `components` of component names, also separated by spaces. The remaining lines specify transitions, and have the following syntax:

```
transition ::= from_state ‘ ’ to_state ‘ ’ action ‘ ’
               feature_expression ‘ ’ components
from_state ::= number
to_state ::= number
action ::= string
feature_expression ::= DNFstring
components ::= string | components ‘,’ components

DNFstring ::= DNFclause | DNFstring ‘|’ DNFstring
DNFclause ::= DNFliteral | DNFclause ‘&’ DNFclause
DNFliteral ::= feature_identifier | ‘!’ feature_identifier
feature_identifier ::= string
```

Here:

- `from_state` is a number representing the state the transition comes from;
- `to_state` is a number representing the state the transition leads to;
- `action` is a string from the set `actions` that represents the action-label of the transition;
- `feature_expression` is a string that represents the feature expression of the transition in disjunctive normal form;
- `components` is a comma separated list representing the component(s) this transition is part of.

Example

A small example FTS file specifying the collective behavior of the vending machine product line is provided on the next page.

```
9
0
pay change free cancel order(Coffee) serve(Coffee)
    order(Water) serve(Water) order(Tea) serve(Tea) take
Machine Beverage Coffee Water Tea FreeDrinks CancelPurchase
0 1 pay m&!f Machine
1 2 change m Machine
6 0 take m Machine
2 3 order(Coffee) c Beverage,Coffee
3 6 serve(Coffee) c Beverage,Coffee
2 4 order(Water) w Beverage,Water
4 6 serve(Water) w Beverage,Water
2 5 order(Tea) t Beverage,Tea
5 6 serve(Tea) t Beverage,Tea
0 2 free f FreeDrinks
2 0 cancel x CancelPurchase
```

F.1.6 FTS-abstraction (.abstr)

An FTS-abstraction specification defines parameters for the FTS-abstraction operator.

Syntax

The syntax of an FTS file consists of a number of lines. The first line contains the set `actions` of action names separated by spaces. the second line contains the set `features` of feature expressions separated by spaces. Feature expressions have the following syntax:

```
feature_expression ::= DNFstring

DNFstring ::= DNFclause | DNFstring '|' DNFstring
DNFclause ::= DNFliteral | DNFclause '&' DNFclause
DNFliteral ::= feature_identifier | '!' feature_identifier
feature_identifier ::= string
```

The third and last line contains the set `components` separated by spaces.
Here:

- `actions` is a list of strings separated by spaces representing the actions that can be abstracted from;
- `features` is a list of feature expression separated by spaces representing feature expression that can not be abstracted from;
- `components` is a list of strings separated by spaces by spaces representing components that can not be abstracted from.

Example

A small example ABSTR file specifying that it is possible to abstract from every action, but not from the `f` feature and the `Beverage` component is provided below.

```
pay change free cancel order(Coffee) serve(Coffee) order(Water)
    serve(Water) order(Tea) serve(Tea) take
f
Beverage
```

F.1.7 SPL-product (.xml)

An SPL-product specification defines a single product of a software product line.

Syntax

The syntax of an SPL-product file consists of an XML schema. An overview of the schema is provided at page 165. Optional XML elements are displayed in square brackets. The schema has an `spl_product` root element, with a mandatory `name` attribute. This attribute specifies the name of the SPL-product. The `spl_product` root element has a number of child elements:

- `labeled_transitions_systems` is an element that specifies the behavior of this SPL-product, as a number of LTS running in parallel. It has at least one `lts` child element with mandatory `name` attribute specifying the path from this SPL-product file to its corresponding LTS file (Section F.1.8).
- `datatypes` is an optional element that specifies user-defined data-types. Each data-type is specified using a `struct` element that has a mandatory `name` attribute specifying the name of the data-type. Each `struct` element has at least on `value` child element with a mandatory `name` attribute, specifying the value name. Note that all values should be unique.
- `parameterized_actions` is an optional element that specifies actions with parameters. Each parameterized action is specified using an `action` element that has a mandatory `name` attribute specifying the name of the action. Its parameters are defined using `parameter` child elements, each having a mandatory attributes `id`, specifying whether this parameter is the first parameter of this actions, the second parameter of this action, etc. Furthermore each parameter has a mandatory `type` attribute specifying the type of the parameter. The value of this parameter can be the name of any user-defined type in the `datatypes` section, or one of the predefined mCRL2 sorts[2] (`Bool`, `Pos`, `Nat`, `Int` or `Real`).
- `communication` is an optional that specifies which multi-actions are renamed to a single action. Each multi-action is specified using a `multiaction` element that has a mandatory `name` attributed specifying the name of the renamed multi-action. The separate actions this multi-actions consists of are specified using `action` child elements, each having a mandatory `name attribute`. All of the actions of a single multi-action must have the same parameter types. Each multi-action element has an optional `type` attribute that should be assigned the value `optional` if communication is not enforced (i.e. the actions of the multi-action can also be executed individually).

Example

A small example SPL-product file specifying product 0 of the vending machine product line is provided on the next page.

[2] `http://www.mcrl2.org/release/user_manual/language_reference/data.html#predefined-sorts`

```
<spl_product name="VendingMachine_SPL_p0">
    <labeled_transition_systems>
        <lts name="VM_FTS.lts" />
    </labeled_transition_systems>
    <datatypes>
        <struct name="Beverage">
            <value name="Coffee" />
            <value name="Tea" />
            <value name="Water" />
        </struct>
    </datatypes>
    <parameterized_actions>
        <action name="order">
            <parameter id="1" type="Beverage" />
        </action>
        <action name="serve">
            <parameter id="1" type="Beverage" />
        </action>
    </parameterized_actions>
</spl_product>
```

The SPL-product file XML schema:

```
<spl_product name="SPL_product_name">

    <labeled_transition_systems>
        <lts name="LTS_file" />
        [ <lts name="LTS_file2" /> ]
        ...
    </labeled_transition_systems>

    [ <datatypes>
        <struct name="Struct1">
            <value name="Value1" />
            [ <value name="Value2" /> ]
            ...
        </struct>
        ...
    </datatypes> ]

    [ <parameterized_actions>
        <action name="Action1">
            <parameter id="1" type="Struct1" />
            [ <parameter id="2" type="Int" /> ]
            ...
        </action>
        ...
    </parameterized_actions> ]

    [ <communication>
        <multiaction name="MultiAction1" [type="optional"]>
            <action name="Act1">
            <action name="Act2">
            [ <action name="Act3"> ]
            ...
        </multiaction>
        ...
    </communication> ]

</spl_product>
```

F.1.8 LTS (.lts)

An LTS specification defines a labeled transition system.

Syntax

The syntax of an LTS file consists of a number of lines. The first line consists of a single number representing the number of states in the FTS. The second line contains the set `actions` of action names separated by spaces. The third line contains the set `components` of component names, also separated by spaces. The remaining lines specify transitions, and have the following syntax:

```
transition ::= from_state to_state action components
from_state ::= number
to_state ::= number
action ::= string
components ::= string | components ‘,’ components
```

Here:

- `from_state` is a number representing the state the transition comes from;
- `to_state` is a number representing the state the transition leads to;
- `action` is a string from the set `actions` that represents the action-label of the transition;
- `components` is a comma separated list representing the component(s) this transition is part of.

Example

A small example LTS file specifying the behavior of product 0 the vending machine product line is provided below.

```
6
0
pay change free cancel order(Coffee) serve(Coffee) order(Water)
    serve(Water) order(Tea) serve(Tea) take
Machine Beverage Coffee Water Tea FreeDrinks CancelPurchase
1 0 take Machine
2 3 order(Coffee) Beverage,Coffee
3 1 serve(Coffee) Beverage,Coffee
2 4 order(Water) Beverage,Water
4 1 serve(Water) Beverage,Water
2 5 order(Tea) Beverage,Tea
5 1 serve(Tea) Beverage,Tea
0 2 free FreeDrinks
```

F.2 Tools

This chapter describes the purpose and usage of each of the eight tools of the FTS Toolset.

F.2.1 parseFD

Parses an SPL file and produces Products and Expressions files.

Usage

```
parseFD -i <INFILE> [-v] [-d]
```

Description

Parse the SPL file specified in INFILE. The input file must be in SPL format. The Products file NAME.prod and Expressions file NAME.expr are produced, where NAME is the SPL-name specified in INFILE. The feature diagram section of INFILE is updated to include the names of the produced files.

Command line options

-iINFILE, –ifileINFILE
parse the FD from INFILE. Must be in SPL format. [**mandatory**]

Standard options

-v, –verbose
display short intermediate messages.

-d, –debug
display detailed intermediate messages.

-h, –help
display help information.

F.2.2 abstractSPL

Produces an abstracted SPL specification.

Usage

```
abstractSPL -i <INFILE> -a <ABSTRFILE> -o <OUTFILE>
```

Description

Abstract the FTS specified in INFILE using the abstraction parameters specified in ABSTRFILE. INFILE must be in SPL format, and ABSTRFILE must be in FTS-abstraction format. The FTS files FTSNAME-abstract.fts are produced, where FTSNAME is the name of an FTS specifed in INFILE. The updated SPL specification is written to OUTFILE, in SPL format.

Command line options

-iINFILE, –ifileINFILE
abstract the FTS from INFILE. Must be in SPL format. [**mandatory**]

-aABSTRFILE, –afileABSTRFILE
use the abstraction parameters from ABSTRFILE. Must be in FTS-abstraction format. [**mandatory**]

-oOUTFILE, –ofileOUTFILE
write the abstracted SPL to OUTFILE. Will be in SPL format. [**mandatory**]

Standard options

-v, –verbose
display short intermediate messages.

-d, –debug
display detailed intermediate messages.

-h, –help
display help information.

F.2.3 BFBreduction

Produces a reduced SPL specification.

Usage

```
BFBreduction -i <INFILE> -o <OUTFILE>
```

Description

Reduce the FTS specified in INFILE using complete coherent branching feature bisimulation reduction. INFILE must be in SPL format. The FTS files FTSNAME-reduced.fts are produced, where FTSNAME is the name of an FTS specifed in INFILE. The updated SPL specification is written to OUTFILE, in SPL format.

Command line options

-iINFILE, –ifileINFILE
reduce the FTS from INFILE. Must be in SPL format. [**mandatory**]

-oOUTFILE, –ofileOUTFILE
write the reduced SPL to OUTFILE. Will be in SPL format. [**mandatory**]

Standard options

-v, –verbose
display short intermediate messages.

-d, –debug
display detailed intermediate messages.

-h, –help
display help information.

F.2.4 spl2mcrl2

Converts and SPL specification to an mCRL2 specification.

Usage

```
spl2mcrl2 -i <INFILE> -o <OUTFILE> [-p]
```

Description

Converts the SPL defined by INFILE to an mCRL2 specification. INFILE must be in SPL format. The produced mCRL2 specification is written to OUTFILE, in mCRL2 format.

Command line options

-iINFILE, –ifileINFILE
convert the SPL from INFILE. Must be in SPL format. [**mandatory**]

-oOUTFILE, –ofileOUTFILE
write the mCRL2 specification to OUTFILE. Will be in mCRL2 format. [**mandatory**]

-p, –parallel
will produce an mCRL2 specification with a process for each component. All components run in parallel and are coordinated by driver processes.

Standard options

-v, –verbose
display short intermediate messages.

-d, –debug
display detailed intermediate messages.

-h, –help
display help information.

F.2.5 fts2aut

Converts an FTS file to an Aldebaran file[3].

Usage

```
fts2aut -i <INFILE> -o <OUTFILE>
```

Description

Converts the FTS defined by INFILE to an LTS specification. INFILE must be in FTS format. The produced LTS specification is written to OUTFILE, in Aldebaran format. Aldebaran format files can be visualized using the tool `ltsgraph`[4] from the mCRL2 toolset. Hence, using a combination of `fts2aut` and `ltsgraph` it is possible to visualize FTS files.

Command line options

-iINFILE, –ifileINFILE
convert the FTS from INFILE. Must be in FTS format. [**mandatory**]

-oOUTFILE, –ofileOUTFILE
write the LTS specification to OUTFILE. Will be in Aldebaran format. [**mandatory**]

Standard options

-v, –verbose
display short intermediate messages.

-d, –debug
display detailed intermediate messages.

-h, –help
display help information.

[3] `http://www.mcrl2.org/release/user_manual/language_reference/lts.html#aldebaran-format`
[4] `http://www.mcrl2.org/release/user_manual/tools/ltsgraph.html`

F.2.6 projectSPL

Project an SPL on a single product.

Usage

```
projectSPL -i <INFILE> -p <NUMBER> -o <OUTFILE>
```

Description

Project the SPL defined by INFILE on the product with number NUMBER. INFILE must be in SPL format. The produced SPL product will be written to OUTFILE, in SPL-product format.

Command line options

-iINFILE, –ifileINFILE
project the SPL from INFILE. Must be in SPL format. [**mandatory**]

-pNUMBER, –productNrNUMBER
project on the product with number NUMBER. [**mandatory**]

-oOUTFILE, –ofileOUTFILE
write the SPL product specification to OUTFILE. Will be in SPL-product format. [**mandatory**]

Standard options

-v, –verbose
display short intermediate messages.

-d, –debug
display detailed intermediate messages.

-h, –help
display help information.

F.2.7 splprod2mcrl2

Converts an SPL-product specification to an mCRL2 specification.

Usage

```
splprod2mcrl2 -i <INFILE> -o <OUTFILE> [-p]
```

Description

Converts the SPL-product defined by INFILE to an mCRL2 specification. INFILE must be in SPL-product format. The produced mCRL2 specification is written to OUTFILE, in mCRL2 format.

Command line options

-iINFILE, –ifileINFILE
convert the SPL-product from INFILE. Must be in SPL-product format. [**mandatory**]

-oOUTFILE, –ofileOUTFILE
write the mCRL2 specification to OUTFILE. Will be in mCRL2 format. [**mandatory**]

-p, –parallel
will produce an mCRL2 specification with a process for each component. All components run in parallel and are coordinated by driver processes.

Standard options

-v, –verbose
display short intermediate messages.

-d, –debug
display detailed intermediate messages.

-h, –help
display help information.

F.2.8 lts2aut

Converts an LTS file to an Aldebaran file[5].

Usage

```
lts2aut -i <INFILE> -o <OUTFILE>
```

Description

Converts the LTS defined by INFILE to an LTS specification. INFILE must be in LTS format. The produced LTS specification is written to OUTFILE, in Aldebaran format. Aldebaran format files can be visualized using the tool `ltsgraph`[6] from the mCRL2 toolset. Hence, using a combination of `fts2aut` and `ltsgraph` it is possible to visualize LTS files.

Command line options

-iINFILE, –ifileINFILE
convert the LTS from INFILE. Must be in LTS format. [**mandatory**]

-oOUTFILE, –ofileOUTFILE
write the LTS specification to OUTFILE. Will be in Aldebaran format. [**mandatory**]

Standard options

-v, –verbose
display short intermediate messages.

-d, –debug
display detailed intermediate messages.

-h, –help
display help information.

[5] http://www.mcrl2.org/release/user_manual/language_reference/lts.html#aldebaran-format
[6] http://www.mcrl2.org/release/user_manual/tools/ltsgraph.html