A Modal Specification Theory for Components with Data

Bauer, Sebastian S.; Larsen, Kim Guldstrand; Legay, Axel; Nyman, Ulrik; Wasowski, Andrzej

Published in:
Formal Aspects of Component Software

DOI (link to publication from Publisher):
10.1007/978-3-642-35743-5_5

Publication date:
2012

Document Version
Accepted author manuscript, peer reviewed version

Link to publication from Aalborg University

Citation for published version (APA):
Bauer, S. S., Larsen, K. G., Legay, A., Nyman, U., & Wasowski, A. (2012). A Modal Specification Theory for Components with Data. In *Formal Aspects of Component Software: 8th International Symposium, FACS 2011, Oslo, Norway, September 14-16, 2011, Revised Selected Papers* (Vol. 7253). Springer Publishing Company. https://doi.org/10.1007/978-3-642-35743-5_5

A Modal Specification Theory for Components with Data[⋆]

Sebastian S. Bauer[1,2], Kim G. Larsen[2], Axel Legay[3], Ulrik Nyman[2], and Andrzej Wąsowski[4]

[1] Institut für Informatik, Ludwig-Maximilians-Universität München, Germany
[2] Department of Computer Science, Aalborg University, Denmark
[3] INRIA/IRISA, Rennes, France
[4] IT University of Copenhagen, Denmark

Abstract. Modal specification is a well-known and widely used formalism used as an abstraction theory for transition systems. Modal specifications are transition systems equipped with two types of transitions: *must*-transitions that are mandatory to any implementation, and *may*-transitions that are optional. The duality of transitions allows to develop a unique approach for both logical and structural compositions, and eases the step-wise refinement process for building implementations.

We propose Modal Specifications with Data (MSD), the first *modal* specification theory with explicit representation of data. Our new theory includes all the essential ingredients of a specification theory. As MSD are potentially infinite-state systems, we propose symbolic representations based on effective predicates. Our theory serves as a new abstraction-based formalism for transition systems with data.

1 Introduction

Modern IT systems are often large and consist of complex assemblies of numerous reactive and interacting components. The components are often designed by independent teams, working under a common agreement on what the interface of each component should be. Consequently, the search for mathematical foundations which support *compositional reasoning* on interfaces is a major research goal. A framework should support inferring properties of the global implementation, designing and advisedly reusing components.

Interfaces are specifications and components that implement an interface are understood as models/implementations. Specification theories should support various features including (1) *refinement*, which allows to compare specifications as well as to replace a specification by another one in a larger design, (2) *structural composition*, which allows to combine specifications of different components, (3) *logical conjunction*, expressing the intersection of the set of requirements expressed by two or more specifications, and last (4) a *quotient operator* that is dual to structural composition and allows synthesizing a component from a set of assumptions.

⋆ Work supported by the German Academic Exchange Service (DAAD), grant D/10/46169, by an “Action de Recherche Collaborative” ARC (TP)I, by MT-LAB, VKR Centre of Excellence, and by the EU project ASCENS, 257414.

Among existing specification theories, one finds modal specifications [1], which are labeled transition systems equipped with two types of transitions: *must*-transitions that are mandatory for any implementation, and *may*-transitions which are optional for an implementation. Modal specifications are known to achieve a more flexible and easy-to-use compositional development methodology for CCS [2], which includes a considerable simplification of the step-wise refinement process proposed by Milner and Larsen. While being very close to logics (conjunction), the formalism takes advantage of a behavioral semantics allowing for easy composition with respect to process construction (structural composition) and synthesis (quotient). However, despite the many advantages, only a few implementations have been considered so far. One major problem is that contrary to other formalisms based on transition systems, there exists no theory of modal specification equipped with rich information such as data variables.

In this paper, we add a new stone to the cathedral of results on modal specifications [3, 4], that is we propose the first such theory equipped with rich data values. Our first contribution is to design a semantical version of modal specifications whose states are split into locations and valuations for possibly infinite-domain variables. For every component, we distinguish between local variables, that are locally controlled by the component, and uncontrolled variables that are controlled by other components and can be accessed, but not modified. Combining variables with sets of actions labeling transitions offers a powerful set of communication primitives that cannot be captured by most existing specification theories. We also propose a symbolic predicate-based representation of our formalism. We consider effective predicates that are closed under conjunction, union, and membership—classical assumptions in existing symbolic theories (e.g. [5]). While the semantic level is possibly infinite-state, the syntactical level permits us to reason on specifications just like one would with the original modal specifications, but with the additional power of rich data.

Continuing our quest, we study modal refinement between specifications. Refinement, which resembles simulation between transition systems, permits to compare sets of implementations in a syntactic manner. Modal refinement is defined at the semantic level, but can also be checked at the symbolic level. We propose a predicate abstraction approach that simplifies the practical complexity of the operation by reducing the number of states and simplifying the predicates. This approach is in line with the work of Godefroid et al. [6], but is applied to specification-based verification rather than to model checking.

We then propose definitions for both logical and structural composition, on the level of symbolic representations of specifications. These definitions are clearly not direct extensions of the ones defined on modal specifications as behaviors of both controlled and uncontrolled variables have to be taken into account. As usual, structural composition offers the property of independent implementability, hence allowing for elegant step-wise refinement. In logical composition, two specifications which disagree on their requirements can be reconciled by synthesizing a new component where conflicts have been removed. This can be done with a symbolic pruning of bad states, which terminates if the system is finite-state, or if the structure of the transition system induced by the specification relies, for instance, on a well-quasi order [7]. Finally, we also propose a quotient operation, that is the dual operation of structural composition, which works for a subclass of systems, and we discuss its limitation. This operator, absent from most existing behavioral and logical specification theories, allows synthesizing a component from a set of assumptions.

In Sect. 2 we introduce modal specifications with data and their finite symbolic representations, refinement, an implementation relation and consistency. In Sect. 3 we define the essential operators of every specification theory, that is parallel composition, conjunction and quotient. For verification of refinement between infinite-state specifications we propose in Sect. 4 an approach based on predicate abstraction techniques. We summarize related works in Sect. 5 and conclude in Sect. 6.

Acknowledgment. We would like to thank Rolf Hennicker for valuable comments on a draft of the paper.

2 Modal Specifications with Data

We will first introduce specifications which are finite symbolic representations of modal specifications with data. We will then propose modal refinement and derive an implementation relation and a consistency notion.

In the following, $\mathscr{P}(M)$ denotes the powerset of M, $\mathscr{P}_{\geq 1}(M) = \mathscr{P}(M) \setminus \{\emptyset\}$, and the union of two disjoint sets is denoted by $M \uplus N$, which is $M \cup N$ with $M \cap N = \emptyset$.

Let $\mathbb{V}$ be a fixed set of variables, each variable ranging over a fixed domain $\mathbb{D}$. For a given subset $V \subseteq \mathbb{V}$, a *data state* s over V is a mapping $s : V \to \mathbb{D}$. If $V = \{x_1, x_2, \ldots, x_n\}$ and $d_1, d_2, \ldots, d_n \in \mathbb{D}$, we write $[x_1 \mapsto d_1, x_2 \mapsto d_2, \ldots, x_n \mapsto d_n]$ for the data state s which maps every x_i to d_i, for $1 \leq i \leq n$. We write $[\![V]\!]$ for the set of all possible data states over V. For disjoint sets of variables V_1 and V_2 and data states $s_1 \in [\![V_1]\!]$ and $s_2 \in [\![V_2]\!]$, the operation $(s_1 \cdot s_2)$ composes the data states resulting in a new state $s = (s_1 \cdot s_2) \in [\![V_1 \uplus V_2]\!]$, such that $s(x) = s_1(x)$ for all $x \in V_1$ and $s(x) = s_2(x)$ for all $x \in V_2$. This is naturally lifted to sets of states: if $S_1 \subseteq [\![V_1]\!]$ and $S_2 \subseteq [\![V_2]\!]$ then $(S_1 \cdot S_2) = \{(s_1 \cdot s_2) \mid s_1 \in S_1, s_2 \in S_2\} \subseteq [\![V_1 \uplus V_2]\!]$.

Like in the work of de Alfaro et al. [8] we define specifications with respect to an assertion language allowing suitable predicate representation. Given a set V of variables, we denote by $\mathit{Pred}(V)$ the set of first-order predicates with free variables in V; we assume that these predicates are written in some specified first-order language with existential ($\exists$) and universal ($\forall$) quantifiers and with interpreted function symbols and predicates; in our examples, the language contains the usual arithmetic operators and boolean connectives ($\vee, \wedge, \neg, \Rightarrow$). Syntactic equality of predicates is written with the symbol $\equiv$. Given a set of variables V we denote by $(V)'$ an isomorphic set of 'primed' variables from V: so if $x \in V$ then $(x)' \in (V)'$. We use this construction to represent pre- and post-values of variables. A variable $(x)' \in (V)'$ represents the next state value of the variable $x \in V$. Given a formula $\varphi \in \mathit{Pred}(V)$ and a data state $s \in [\![V]\!]$, we write $\varphi(s)$ if the predicate formula φ is true when its free variables are interpreted as specified by s. Given a formula $\psi \in \mathit{Pred}(V_1 \uplus (V_2)')$ and states $s_1 \in [\![V_1]\!]$, $s_2 \in [\![V_2]\!]$, we often write $\psi(s_1, s_2)$ for $\psi(s_1 \cdot t_2)$ where $t_2 \in [\![(V_2)']\!]$ such that $t_2((x)') = s_2(x)$ for all $x \in V_2$. Given a predicate $\varphi \in \mathit{Pred}(V)$, we write $(\varphi)' \in \mathit{Pred}((V)')$ for the predicate obtained by substituting x with $(x)'$ in φ, for all $x \in V$. We write $[\![\varphi]\!]$ for the set $\{s \in [\![V]\!] \mid \varphi(s)\}$ which consists of all states satisfying $\varphi \in \mathit{Pred}(V)$ (for predicates with primed and unprimed variables), and φ is *consistent* if $[\![\varphi]\!] \neq \emptyset$. We write $\exists V \varphi$ meaning existential quantification of φ over all variables in the set V, and similar for universal quantification. Finally, for a predicate $\psi \in \mathit{Pred}(V_1 \uplus (V_2)')$, we write ${}^\circ\psi$ for $\exists (V_2)'\psi$, and ψ° for $\exists V_1 \psi$.

Our theory enriches modal automata with variables. Specifications not only express constraints on the allowed sequences of actions, but also their dependence and effect on the values of variables. Like in the loose approach of modal specifications [1] which allows under-specification using *may* and *must* modalities on transitions, we allow loose specification of the effects of actions on the data state. From a given location and a given data state, a transition to another location is allowed to lead to several next data states. Unlike in modal specifications, variables are observable in our framework, allowing for modeling shared variable communication.

A *signature* $\mathit{Sig} = (\Sigma, V^L, V^G)$ determines the alphabet of actions Σ and the set of variables $V = V^L \uplus V^G$ of an interface. The variables in V^L are *local (controlled) variables*, owned by the interface and visible to any other component. V^G contains the *uncontrolled variables* owned by the environment, which are read-only for the interface.

Specifications are finite modal transition systems where transitions are equipped with predicates. A transition predicate $\psi \in \mathit{Pred}(V \uplus (V^L)')$ relates a previous state, determined by all controlled and uncontrolled data states, with the next possible controlled data state.

Definition 1. *A* specification *is a tuple* $\mathbf{A} = (\mathit{Sig}, \mathit{Loc}, \ell^0, \varphi^0, E_\Diamond, E_\Box)$ *where* $\mathit{Sig} = (\Sigma, V^L, V^G)$ *is a signature, Loc is a finite set of locations,* $\ell^0 \in \mathit{Loc}$ *is the initial location,* $\varphi^0 \in \mathit{Pred}(V^L)$ *is a predicate on the initial local state, and* $E_\Diamond$*,* $E_\Box$ *are finite may- and must-transition relations respectively:*

$$E_\Diamond, E_\Box \subseteq \mathit{Loc} \times \Sigma \times \mathit{Pred}(V \uplus (V^L)') \times \mathit{Loc}.$$

Given a specification $\mathbf{A}$, locations $\ell, \ell' \in \mathit{Loc}$, and action $a \in \Sigma$, we refer to the set of transition predicates on may-transitions by $\mathit{May}^a(\ell, \ell') = \{\psi \mid (\ell, a, \psi, \ell') \in E_\Diamond\}$ and on must-transitions by $\mathit{Must}^a(\ell, \ell') = \{\psi \mid (\ell, a, \psi, \ell') \in E_\Box\}$.

Example 1. Consider a specification of a print server, shown in Fig. 1. Must-transitions are drawn with solid arrows and may-transitions with dashed ones. Every solid arrow representing a must-transition has an implicit may-transition shadowing it which is not shown. Every transition is equipped with a transition predicate over unprimed variables, referring to the pre-state, and primed variables, referring to the poststate. The print server receives new print jobs ($\mathbf{newPrintJob}$), stores them and assigns them either a low or high priority; the numbers of low and high priority jobs are modeled by controlled variables l and h, respectively; l and h are natural numbers. A job with low priority can also be reclassified to high priority ($\mathbf{incPriority}$). The printer server can send ($\mathbf{send}$) a job to a printer, and then wait for the acknowledgment ($\mathbf{ack}$). In state ℓ_1, if there is a job with high priority and the uncontrolled boolean variable $\mathit{priorityMode}$ is true, then there must be a send transition. The specification is loose in the sense that if a second print job is received in state ℓ_1, then the behavior is left unspecified.

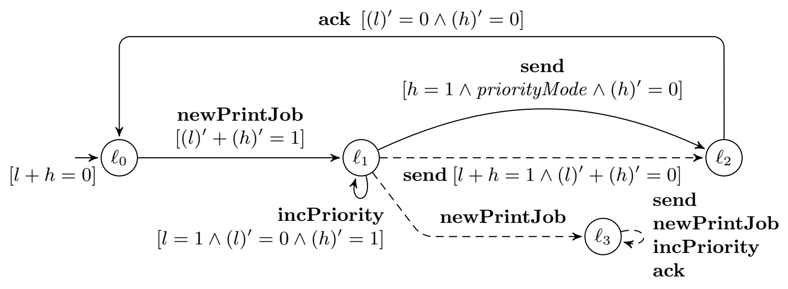
ack $[(l)'=0 \wedge (h)'=0]$
send
$[h=1 \wedge \mathit{priorityMode} \wedge (h)'=0]$
newPrintJob
$[(l)'+(h)'=1]$
ℓ_0
$[l+h=0]$
ℓ_1
ℓ_2
send $[l+h=1 \wedge (l)'+(h)'=0]$
incPriority
$[l=1 \wedge (l)'=0 \wedge (h)'=1]$
newPrintJob
ℓ_3
send
newPrintJob
incPriority
ack

Fig. 1. Abstract specification $\mathbf{P}$ of a print server.

We now define the kind of transition systems which will be used for formalizing the semantics of specifications. A specification is interpreted as a variant of modal transition systems where the *state space* is formed by the cartesian product $\mathit{Loc} \times [\![V^L]\!]$, i.e. a *state* is a pair (ℓ, s) where $\ell \in \mathit{Loc}$ is a location and $s \in [\![V^L]\!]$ is a valuation of the controlled variables. To motivate the choice of the transition relations in the semantics of specifications, we first describe the intended meaning of may- and must-transitions.

A may-transition $(\ell, a, \psi, \ell') \in E_\Diamond$ in the specification expresses that in any implementation, in any state (ℓ, s) and for any guard $g \in [\![V^G]\!]$ (that is a valuation of uncontrolled variables V^G) the implementation is *allowed* to have a transition with guard g and action a to a next state (ℓ', s') such that $\psi(s \cdot g, s')$. The interpretation of a must-transition $(\ell, a, \psi, \ell') \in E_\Box$ is a bit more involved: Any implementation, in state (ℓ, s), and for any guard $g \in [\![V^G]\!]$, if there is a valuation $s' \in [\![V^L]\!]$ such that $\psi(s \cdot g, s')$, then the implementation is *required* to have a transition from state (ℓ, s) with guard g and action a to *at least some* state t' such that $\psi(s \cdot g, t')$. The requirement expressed by must-transitions cannot be formalized by standard modal transition systems, but fortunately, a generalization called disjunctive modal transition systems introduced in [9] can precisely capture these requirements. May-transitions target (as usual) only one state, but must-transitions branch to several possible next states (thus must-transitions are hypertransitions), with an existential interpretation: there must exist at least one transition with some target state which is an element from the set of target states of the hypertransition.

Definition 2. *A* modal specification with data (MSD) *is a tuple*

$$\mathbf{S} = (\mathit{Sig}, \mathit{Loc}, \ell^0, S^0, \longrightarrow_\Diamond, \longrightarrow_\Box)$$

where Sig, Loc, ℓ^0 are like in Def. 1, $S^0 \subseteq [\![V^L]\!]$ is a set of initial data states, and $\longrightarrow_\Diamond, \longrightarrow_\Box \subseteq \mathit{Loc} \times [\![V^L]\!] \times [\![V^G]\!] \times \Sigma \times (\mathit{Loc} \times \mathscr{P}_{\geq 1}([\![V^L]\!]))$ are the may- ($\Diamond$) and must- ($\Box$) transition relations such that every may-transition targets a single state: if $(\ell, s, g, a, (\ell', S')) \in \longrightarrow_\Diamond$ then $|S'| = 1$.

A state $(\ell, s) \in \mathit{Loc} \times [\![V^L]\!]$ is called syntactically consistent *iff targets reachable by must-transitions are also reachable by may-transitions: if $(\ell, s, g, a, (\ell', S')) \in \longrightarrow_\Box$ then $(\ell, s, g, a, (\ell', \{s'\})) \in \longrightarrow_\Diamond$ for all $s' \in S'$.* $\mathbf{S}$ *is* syntactically consistent *iff all states are syntactically consistent, and the set of initial data states is nonempty, i.e. $S^0 \neq \emptyset$.*

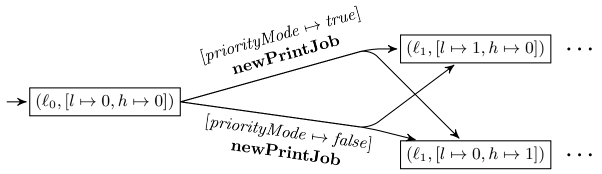
$(\ell_0, [l \mapsto 0, h \mapsto 0])$
$[priorityMode \mapsto true]$
$\mathbf{newPrintJob}$
$[priorityMode \mapsto false]$
$\mathbf{newPrintJob}$
$(\ell_1, [l \mapsto 1, h \mapsto 0])$
$(\ell_1, [l \mapsto 0, h \mapsto 1])$
$\cdots$
$\cdots$

Fig. 2. Excerpt of the semantics of the abstract print server specification.

May-transitions $(\ell, s, g, a, (\ell', S')) \in \longrightarrow_\Diamond$ are often written $(\ell, s) \xrightarrow{g\,a}_\Diamond (\ell', S')$, and similarly for must-transitions.

We can now define formally how a specification translates to its semantics in terms of an MSD. As already described above, the semantics of a may-transition of the specification is given by the set of may-transitions pointing to single admissible target states, and a must-transition gives rise to (must-)hypertransitions targeting all the admissible poststates.

Definition 3. *The* semantics *of a specification* $\mathbf{A} = (Sig, Loc, \ell^0, \varphi^0, E_\Diamond, E_\Box)$ *is given by the MSD* $\langle \mathbf{A} \rangle_{\mathrm{sem}} = (Sig, Loc, \ell^0, S^0, \longrightarrow_\Diamond, \longrightarrow_\Box)$ *where* $S^0 = [\![\varphi^0]\!]$ *and the transition relations are defined as follows. For each* $\ell, \ell' \in Loc$, $s, s' \in [\![V^L]\!]$, $g \in [\![V^G]\!]$, *and* $a \in \Sigma$*:*

i. If $(\ell, a, \psi, \ell') \in E_\Diamond$ *and* $\psi(s \cdot g, s')$ *then* $(\ell, s) \xrightarrow{g\,a}_\Diamond (\ell', \{s'\})$*,*
ii. If $(\ell, a, \psi, \ell') \in E_\Box$ *and* $\psi(s \cdot g, s')$ *then* $(\ell, s) \xrightarrow{g\,a}_\Box (\ell', \{t' \in [\![V^L]\!] \mid \psi(s \cdot g, t')\})$*.*

A specification $\mathbf{A}$ is called *syntactically consistent* iff its semantics $\langle \mathbf{A} \rangle_{\mathrm{sem}}$ is syntactically consistent. In the following we will always assume that specifications and MSD are syntactically consistent.

Example 2. An excerpt of the semantics of our abstract specification of the print server (see Fig. 1) can be seen Fig. 2. As before, we draw must-transitions with a solid arrow, and has an implicit set of may-transitions shadowing it which are not shown, i.e. for each target (ℓ, S') of a must-transition and each $s \in S'$ there is a may-transition with the same source state and with target state $(\ell, \{s\})$.

The first must-transition $(\ell_0, \mathbf{newPrintJob}, (l)' + (h)' = 1, \ell_1) \in E_\Box$ of the print server specification gives rise to the transitions shown in Fig. 2. Any new print job must be stored in either l or h but which one is not yet fixed by the specification. Thus in the semantics this is expressed as a disjunctive must-transition to the unique location ℓ_1 and the next possible data states $[l \mapsto 1, h \mapsto 0]$ and $[l \mapsto 0, h \mapsto 1]$.

A *refinement relation* allows to relate a concrete specification with an abstract specification. Refinement should satisfy the following substitutability property: If $\mathbf{A}$ refines $\mathbf{B}$ then replacing $\mathbf{B}$ with $\mathbf{A}$ in a context $\mathcal{C}[\cdot]$ gives a specification $\mathcal{C}[\mathbf{A}]$ refining $\mathcal{C}[\mathbf{B}]$. Refinement will be a precongruence, i.e. it is compatible with the structural and logical operators on specifications in the above sense.

Our definition of refinement is based on modal refinement [10, 9] for (disjunctive) modal transition systems, where the may-transitions determine which actions are permitted in a refinement while the must-transitions specify which actions must be present in a refinement and hence in any implementation. We adapt it with respect to data states.

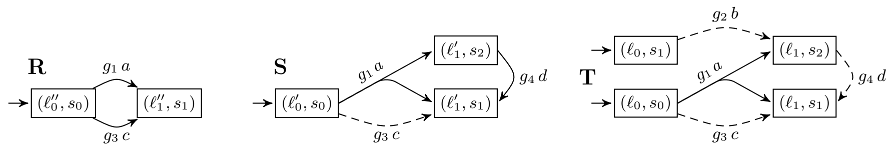

Fig. 3. Successive refinement of an MSD $\mathbf{T}$.

Example 3. We motivate our adaption of modal refinement to take into account data states with the help of a small example shown in Fig. 3. We draw may-transitions with a dashed arrow, and must-transitions with a solid arrow. Every must-transition has an implicit set of may-transitions shadowing it which are not shown. The MSD $\mathbf{T}$ (to the right) has two initial states, both having ℓ_0 as the initial location. The must-transition starting from (ℓ_0, s_0) expresses that in any implementation there must be a transition leading to at least one of the states (ℓ_1, s_1) and (ℓ_1, s_2). The MSD $\mathbf{T}$ can be refined to the MSD $\mathbf{S}$ (by dropping one may-transition and turning one may-transition to a must-transition), and then $\mathbf{S}$ is refined by the MSD $\mathbf{R}$, by refining the must-transition $(\ell_0', s_0, g_1, a, (\ell_1', \{s_1, s_2\}))$ in $\mathbf{S}$ to the must-transition $(\ell_0'', s_0, g_1, a, (\ell_1'', \{s_1\}))$ in $\mathbf{R}$, and by strengthening the transition with guard g_3 and action c to a must-transition.

Definition 4. *Let* $\mathbf{T}_1 = (\mathit{Sig}, \mathit{Loc}_1, \ell_1^0, S_1^0, \longrightarrow_{\Diamond,1}, \longrightarrow_{\Box,1})$ *and* $\mathbf{T}_2 = (\mathit{Sig}, \mathit{Loc}_2, \ell_2^0, S_2^0, \longrightarrow_{\Diamond,2}, \longrightarrow_{\Box,2})$ *be MSD over the same signature* $\mathit{Sig} = (\Sigma, V^L, V^G)$*. A relation* $R \subseteq \mathit{Loc}_1 \times \mathit{Loc}_2 \times [\![V^L]\!]$ *is a* refinement relation *iff for all* $(\ell_1, \ell_2, s) \in R$*:*

i. Whenever $(\ell_1, s) \xrightarrow{g\, a}_{\Diamond,1} (\ell_1', \{s'\})$ *then there exists* $(\ell_2, s) \xrightarrow{g\, a}_{\Diamond,2} (\ell_2', \{t'\})$ *such that* $s' = t'$ *and* $(\ell_1', \ell_2', s') \in R$*.*
ii. Whenever $(\ell_2, s) \xrightarrow{g\, a}_{\Box,2} (\ell_2', S_2')$ *then there exists* $(\ell_1, s) \xrightarrow{g\, a}_{\Box,1} (\ell_1', S_1')$ *such that* $S_1' \subseteq S_2'$ *and* $(\ell_1', \ell_2', s') \in R$ *for all* $s' \in S_1'$*.*

We say that $\mathbf{T}_1$ refines $\mathbf{T}_2$*, written* $\mathbf{T}_1 \leq_{\text{sem}} \mathbf{T}_2$*, iff* $S_1^0 \subseteq S_2^0$ *and there exists a refinement relation* R *such that for any* $s \in S_1^0$ *also* $(\ell_1^0, \ell_2^0, s) \in R$*. A specification* $\mathbf{A}_1$ *refines another specification* $\mathbf{A}_2$*, written* $\mathbf{A}_1 \leq \mathbf{A}_2$*, iff* $\langle \mathbf{A}_1 \rangle_{\text{sem}} \leq_{\text{sem}} \langle \mathbf{A}_2 \rangle_{\text{sem}}$*.*

The refinement relation is a preorder on the class of all specifications. Refinement can be checked in polynomial time in the size of the state space of the MSD (for variables with finite domains). In general the domain may be infinite, or prohibitively large, so in Sect. 4 we revisit the question of refinement checking using abstraction techniques.

Example 4. The semantics of our abstract print server specification, shown in Fig. 2, can be refined as shown in Fig. 4. Now, both must-transitions point to the location ℓ_1 with the data state $[l \mapsto 1, h \mapsto 0]$ which means that any new incoming print job is assigned a low priority, independent of the uncontrolled variable $\mathit{priorityMode}$.

An MSD for which the conditions (1) $\longrightarrow_{\Diamond} = \longrightarrow_{\Box}$ and (2) $|S^0| = 1$ are satisfied, can be interpreted as (an abstraction of) an *implementation*: there are no design choices left open as (1) all may-transitions are covered by must-transitions and (2) there is only one initial data state possible. Any MSD for which the conditions (1) and (2) are satisfied, is called *transition system with data (TSD)* in the following. Note that TSD cannot be

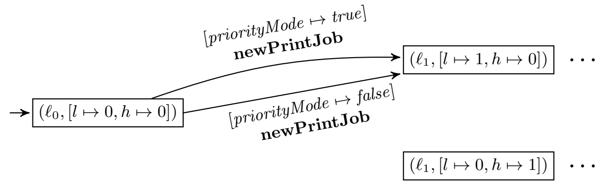
$[\mathit{priorityMode} \mapsto \mathit{true}]$
newPrintJob
$(\ell_1, [l \mapsto 1, h \mapsto 0])$
$\cdots$
$(\ell_0, [l \mapsto 0, h \mapsto 0])$
$[\mathit{priorityMode} \mapsto \mathit{false}]$
newPrintJob
$(\ell_1, [l \mapsto 0, h \mapsto 1])$
$\cdots$

Fig. 4. Refinement of the MSD shown in Fig. 2.

strictly refined, i.e. for any TSD $\mathbf{I}$ and any MSD $\mathbf{S}$ with the same signature, $\mathbf{S} \leq_{\mathrm{sem}} \mathbf{I}$ implies $\mathbf{I} \leq_{\mathrm{sem}} \mathbf{S}$.

An implementation relation connects specifications to implementations (given as TSD) satisfying them. We can simply use refinement as the implementation relation. Given a specification $\mathbf{A}$ and some TSD $\mathbf{I}$, we write $\mathbf{I} \models \mathbf{A}$ for $\mathbf{I} \leq_{\mathrm{sem}} \langle \mathbf{A} \rangle_{\mathrm{sem}}$, so our implementation $\mathbf{I}$ is seen as the model which satisfies the property expressed by the specification $\mathbf{A}$. Now the set of implementations of a specification is the set of all its refining TSD: given a specification $\mathbf{A}$, we define $\mathit{Impl}(\mathbf{A}) = \{\mathbf{I} \mid \mathbf{I} \models \mathbf{A}\}$.

Our implementation relation $\models$ immediately leads to the classical notion of consistency as existence of models. A specification $\mathbf{A}$ is *consistent* iff $\mathit{Impl}(\mathbf{A})$ is non-empty. Consequently, as modal refinement is reflexive, any specification $\mathbf{A}$ for which $\langle \mathbf{A} \rangle_{\mathrm{sem}}$ is a TSD, is consistent.

By transitivity, modal refinement entails implementation set inclusion: for specifications $\mathbf{A}$ and $\mathbf{B}$, if $\mathbf{A} \leq \mathbf{B}$ then $\mathit{Impl}(\mathbf{A}) \subseteq \mathit{Impl}(\mathbf{B})$. The relation $\mathit{Impl}(\mathbf{A}) \subseteq \mathit{Impl}(\mathbf{B})$ is sometimes called *thorough refinement* [11]. Just like for modal transition systems, thorough refinement does not imply modal refinement in general [12]. To establish equivalence we follow [13] by imposing a restriction on $\mathbf{B}$, namely that it is deterministic. An MSD is *deterministic* if

(1) if $(\ell, s, g, a, (\ell', S')), (\ell, s, g, a, (\ell'', S'')) \in \longrightarrow_{\Box}$ then $(\ell', S') = (\ell'', S'')$,
(2) if $(\ell, s, g, a, (\ell', S')), (\ell, s, g, a, (\ell'', S'')) \in \longrightarrow_{\Diamond} \cup \longrightarrow_{\Box}$ then $\ell' = \ell''$.

A specification $\mathbf{B}$ is *deterministic*, if the MSD $\langle \mathbf{B} \rangle_{\mathrm{sem}}$ is deterministic. Note that for may-transitions, determinism only requires that for the same source state, guard and action, the transition leads to a unique next location. The reason why this is sufficient is that modal refinement explicitely distinguishes states by their data state part: two states (ℓ, s) and (ℓ', s') can only be related if their data state parts s, s' coincide.

Now, turning back to the relationship of modal refinement and inclusion of implementation sets (thorough refinement), we can prove the following theorem. Under the restriction of determinism of the refined (abstract) specification we can prove completeness of refinement. This theorem effectively means that modal refinement, as defined for MSD, is characterized by set inclusion of admitted implementations.

Theorem 1. *Let $\mathbf{A}$ and $\mathbf{B}$ be two specifications with the same signature such that $\mathbf{B}$ is deterministic. Then $\mathbf{A} \leq \mathbf{B}$ if and only if $\mathit{Impl}(\mathbf{A}) \subseteq \mathit{Impl}(\mathbf{B})$.*

3 Compositional Reasoning

In this section we propose all the essential operators on specifications a good specification theory should provide. We will distinguish between structural and logical composition.

Structural composition mimics the classical composition of transition systems at the specification level. Logical composition allows to compute the intersection of sets of models and hence can be used to represent the conjunction of requirements made on an implementation. Furthermore we will introduce a quotient operator which is the dual operator to structural composition.

From now on, we assume that for any two specifications with the signatures $Sig_1 = (\Sigma_1, V_1^L, V_1^G)$ and $Sig_2 = (\Sigma_2, V_2^L, V_2^G)$, respectively, we can assume that $\Sigma_1 = \Sigma_2$ and $V_1^L \uplus V_1^G = V_2^L \uplus V_2^G$. This is not a limitation, as one can apply the constructions of [4] to equalize alphabets of actions and sets of variables.

Parallel composition. Two specifications $\mathbf{A}_1$ and $\mathbf{A}_2$ with $Sig_1 = (\Sigma_1, V_1^L, V_1^G)$, $Sig_2 = (\Sigma_2, V_2^L, V_2^G)$, respectively, are *composable* iff $V_1^L \cap V_2^L = \emptyset$. Then their signatures can be composed in a straightforward manner to the signature

$$Sig_1 \parallel Sig_2 = (\Sigma_1, V_1^L \cup V_2^L, (V_1^G \cup V_2^G) \setminus (V_1^L \cup V_2^L))$$

in which the set of controlled variables is the union of the sets of controlled variables of $\mathbf{A}_1$ and $\mathbf{A}_2$, and the set of uncontrolled variables consists of all those uncontrolled variables of $\mathbf{A}_1$ and $\mathbf{A}_2$ which are controlled neither by $\mathbf{A}_1$ nor by $\mathbf{A}_2$.

Definition 5. *Let $\mathbf{A}_1$ and $\mathbf{A}_2$ be two composable specifications. The* parallel composition *of $\mathbf{A}_1$ and $\mathbf{A}_2$ is defined as the specification*

$$\mathbf{A}_1 \parallel \mathbf{A}_2 = (Sig_1 \parallel Sig_2, Loc_1 \times Loc_2, (\ell_1^0, \ell_2^0), \varphi_1^0 \wedge \varphi_2^0, E_\Diamond, E_\Box)$$

where the transition relations $E_\Diamond$ and $E_\Box$ are the smallest relations satisfying the rules:

1. *if $(\ell_1, a, \psi_1, \ell_1') \in E_{\Diamond,1}$ and $(\ell_2, a, \psi_2, \ell_2') \in E_{\Diamond,2}$ then $((\ell_1, \ell_2), a, \psi_1 \wedge \psi_2, (\ell_1', \ell_2')) \in E_\Diamond$,*
2. *if $(\ell_1, a, \psi_1, \ell_1') \in E_{\Box,1}$ and $(\ell_2, a, \psi_2, \ell_2') \in E_{\Box,2}$ then $((\ell_1, \ell_2), a, \psi_1 \wedge \psi_2, (\ell_1', \ell_2')) \in E_\Box$.*

Composition of specifications, similar to the classical notion of modal composition for modal transition systems [10], synchronizes on matching shared actions and only yields a must-transition if there exist corresponding matching must-transitions in the original specifications. Composition is commutative (up to isomorphism) and associative. Our theory supports independent implementability of specifications, which is a crucial requirement for any compositional specification framework [14].

Theorem 2. *Let $\mathbf{A}_1, \mathbf{A}_2, \mathbf{B}_1, \mathbf{B}_2$ be specifications such that $\mathbf{A}_1$ and $\mathbf{B}_1$ are composable. If $\mathbf{A}_1 \leq \mathbf{A}_2$ and $\mathbf{B}_1 \leq \mathbf{B}_2$, then $\mathbf{A}_1 \parallel \mathbf{B}_1 \leq \mathbf{A}_2 \parallel \mathbf{B}_2$.*

The analog of parallel composition on the level of specifications is parallel composition $\parallel_{\text{sem}}$ on the level of MSD which is a straightforward translation of the above symbolic rules. In fact one can prove that both parallel compositions $\parallel$ and $\parallel_{\text{sem}}$ are equivalent, i.e. that $\langle \mathbf{A}_1 \parallel \mathbf{A}_2 \rangle_{\text{sem}} = \langle \mathbf{A}_1 \rangle_{\text{sem}} \parallel_{\text{sem}} \langle \mathbf{A}_2 \rangle_{\text{sem}}$ for any two composable specifications $\mathbf{A}_1, \mathbf{A}_2$.

Remark 1. Interface theories based on transition systems labeled with input/output actions usually involve a notion of compatibility, which is a relation between interfaces determining whether two components can work properly together. Since the present theory does not have a notion of input/output it is enough to require that two components are composable, i.e. that their local variables do not overlap. A pessimistic input/output compatibility notion has been proposed in our previous work [15]. Optimistic input/output compatibility based on a game semantics allows computing all the environments in which two components can work together. Following our recent works in [16, 4], one can enrich labels of transitions in the present theory with input and output and apply the same game-based semantics in order to achieve an optimistic composition.

Syntactical consistency. Our next two specification operators, conjunction and quotient, may yield specifications which are *syntactically inconsistent*, i.e. either there is no legal initial data state or there are states with a must-transition but without corresponding may-transition.

In general, given a specification $\mathbf{A}$, syntactic consistency implies consistency, i.e. $\mathit{Impl}(\mathbf{A}) \neq \emptyset$, but in general, the reverse does not hold. However, every consistent specification can be "pruned" to a syntactically consistent one, by pruning backwards from all syntactically inconsistent states, removing states which have to reach some of the "bad" states. Pruning will be shown to preserve the set of implementations.

For a specification $\mathbf{A} = (\mathit{Sig}, \mathit{Loc}, \ell^0, \varphi^0, E_\Diamond, E_\Box)$, the pruning (or reduction) of $\mathbf{A}$, denoted by $\rho(\mathbf{A})$, is done as follows. Let $B : \mathit{Loc} \to \mathit{Pred}(V^L)$ be a mapping of locations to predicates over the local variables. We define a predecessor operation, iteratively computing all states that are forced to reach a "bad" state. Define a weakest precondition predicate, for $\psi \in \mathit{Pred}(V \uplus (V^L)')$, $\varphi \in \mathit{Pred}(V^L)$, by

$$\mathsf{wp}_\psi[\varphi] \equiv \exists V^G.{}^\circ\psi \wedge (\forall (V^L)'.\psi \Rightarrow (\varphi)') \tag{1}$$

which computes the largest set of local states such that there exists an uncontrolled state $g \in [\![V^G]\!]$ such that ψ maps to at least one next state, and all next states satisfy φ. Then

$$\mathsf{predec}(B)(\ell) \equiv B(\ell) \vee \bigvee\nolimits_{a\in\Sigma,\ell'\in\mathit{Loc},\psi\in\mathit{Must}^a(\ell,\ell')} \mathsf{wp}_\psi[B(\ell')]$$

and $\mathsf{predec}^0(B) \equiv B$, $\mathsf{predec}^{j+1}(B) \equiv \mathsf{predec}(\mathsf{predec}^j(B))$ for $j \geq 0$, and then finally $\mathsf{predec}^*(B) \equiv \bigcup_{j\geq 0} \mathsf{predec}^j(B)$. Define $\mathsf{bad} : \mathit{Loc} \to \mathit{Pred}(V^L)$, for any $\ell \in \mathit{Loc}$, by

$$\mathsf{bad}(\ell) \equiv \bigvee_{a\in\Sigma,\ell'\in\mathit{Loc},\psi\in\mathit{Must}^a(\ell,\ell')} \exists V^G.{}^\circ\psi \wedge \left(\forall (V^L)'.\psi \Rightarrow \bigwedge_{\psi'\in\mathit{May}^a(\ell,\ell')} \neg\psi'\right)$$

and thus $\mathsf{bad}(\ell)$ is satisfied by a valuation $s \in [\![V^L]\!]$ iff there is a must-transition for which no choice of the next data state is permitted by the may-transitions.

In general, for infinite-domain variables, the computation of $\mathsf{predec}^*(\mathsf{bad})$ may not terminate. In [7], it was shown that reachability and related properties in well-structured transition systems with data values, that are monotonic transition systems with a well-quasi ordering on the set of data values, is decidable. This result can be used for specifications with infinite-domain variables to show that under these assumptions, there is some $j \geq 0$ such that for all $\ell \in \mathit{Loc}$, $[\![\mathsf{predec}^j(\mathsf{bad})(\ell)]\!] = [\![\mathsf{predec}^{j+1}(\mathsf{bad})(\ell)]\!]$. In the following, for the specification operators conjunction and quotient (which may result in a syntactically inconsistent specification and hence need to be pruned) we assume that such a $j \geq 0$ exists.

The *pruning* $\rho(\mathbf{A})$ of $\mathbf{A}$ is defined if $\varphi^0 \land \neg\mathsf{predec}^j(\mathsf{bad})(\ell^0)$ is consistent; and in this case, $\rho(\mathbf{A})$ is the specification $(\mathit{Sig}, \mathit{Loc}, \ell^0, \varphi^0 \land \neg\mathsf{predec}^j(\mathsf{bad})(\ell^0), E^\rho_\Diamond, E^\rho_\Box)$ where, for $\chi_{good} = \neg\mathsf{predec}^j(\mathsf{bad})$,

$$\begin{aligned} E^\rho_\Diamond &= \big\{(\ell_1, a, \chi_{good}(\ell_1) \land \psi \land (\chi_{good}(\ell_2))', \ell_2) \mid (\ell_1, a, \psi, \ell_2) \in E_\Diamond\big\}, \\ E^\rho_\Box &= \big\{(\ell_1, a, \chi_{good}(\ell_1) \land \psi \land (\chi_{good}(\ell_2))', \ell_2) \mid (\ell_1, a, \psi, \ell_2) \in E_\Box\big\}. \end{aligned}$$

Crucially the pruning operator has the expected properties:

Theorem 3. *Let $\mathbf{A}$ be a deterministic, possibly syntactically inconsistent specification. Then $\rho(\mathbf{A})$ is defined if and only if $\mathbf{A}$ is consistent. And if $\rho(\mathbf{A})$ is defined, then*

1. *$\rho(\mathbf{A})$ is a specification (hence syntactically consistent),*
2. *$\rho(\mathbf{A}) \leq \mathbf{A}$,*
3. *$\mathit{Impl}(\mathbf{A}) = \mathit{Impl}(\rho(\mathbf{A}))$, and*
4. *for any specification $\mathbf{B}$, if $\mathbf{B} \leq \mathbf{A}$, then $\mathbf{B} \leq \rho(\mathbf{A})$.*

Logical composition. Conjunction of two specifications yields the greatest lower bound with respect to modal refinement. Syntactic inconsistencies arise if one specification requires a behavior disallowed by the other.

Definition 6. *Let $\mathbf{A}_1$ and $\mathbf{A}_2$ be two specifications with the same signature Sig. The* conjunction of $\mathbf{A}_1$ and $\mathbf{A}_2$ *is defined as the possibly syntactically inconsistent specification*

$$\mathbf{A}_1 \land \mathbf{A}_2 = (\mathit{Sig}, \mathit{Loc}_1 \times \mathit{Loc}_2, (\ell^0_1, \ell^0_2), \varphi^0_1 \land \varphi^0_2, E_\Diamond, E_\Box)$$

where the transition relations $E_\Diamond$, $E_\Box$ are the smallest relations satisfying the rules, for any $\ell_1 \in \mathit{Loc}_1$, $\ell_2 \in \mathit{Loc}_2$, $a \in \Sigma$,

1. *If $(\ell_1, a, \psi_1, \ell'_1) \in E_{\Diamond,1}$, $(\ell_2, a, \psi_2, \ell'_2) \in E_{\Diamond,2}$, then* $((\ell_1, \ell_2), a, \psi_1 \land \psi_2, (\ell'_1, \ell'_2)) \in E_\Diamond$,
2. *If $(\ell_1, a, \psi_1, \ell'_1) \in E_{\Box,1}$, then* $((\ell_1, \ell_2), a, \psi_1 \land (\bigvee_{\psi_2 \in \mathit{May}^a_2(\ell_2, \ell'_2)} \psi_2), (\ell'_1, \ell'_2)) \in E_\Box$,
3. *If $(\ell_2, a, \psi_2, \ell'_2) \in E_{\Box,2}$, then* $((\ell_1, \ell_2), a, \psi_2 \land (\bigvee_{\psi_1 \in \mathit{May}^a_1(\ell_1, \ell'_1)} \psi_1), (\ell'_1, \ell'_2)) \in E_\Box$,
4. *If $(\ell_1, a, \psi_1, \ell'_1) \in E_{\Box,1}$ then* $((\ell_1, \ell_2), a, {}^\circ\psi_1 \land \Big(\forall (V^L)'. \psi_1 \Rightarrow \bigwedge_{\psi_2 \in M} \neg\psi_2\Big), (\ell'_1, \ell_2)) \in E_\Box$, *where $M = \bigcup_{\ell'_2 \in \mathit{Loc}_2} \mathit{May}^a_2(\ell_2, \ell'_2)$,*
5. *If $(\ell_2, a, \psi_2, \ell'_2) \in E_{\Box,2}$ then* $((\ell_1, \ell_2), a, {}^\circ\psi_2 \land \Big(\forall (V^L)'. \psi_2 \Rightarrow \bigwedge_{\psi_1 \in M} \neg\psi_1\Big), (\ell_1, \ell'_2)) \in E_\Box$, *where $M = \bigcup_{\ell'_1 \in \mathit{Loc}_1} \mathit{May}^a_1(\ell_1, \ell'_1)$.*

The first rule composes may-transitions (with the same action) by conjoining their predicates. Rule (2) and (3) express that any required behavior of $\mathbf{A}_1$ ($\mathbf{A}_2$ resp.), as long as it is allowed by $\mathbf{A}_2$ ($\mathbf{A}_1$ resp.), is also a required behavior in $\mathbf{A}_1 \wedge \mathbf{A}_2$. Rules (4) and (5) capture the case when a required behavior of $\mathbf{A}_1$ is not allowed by $\mathbf{A}_2$. Conjunction is commutative and associative.

Refinement is a precongruence with respect to conjunction for deterministic specifications. Moreover, under the assumption of determinism, the conjunction construction yields the greatest lower bound with respect to modal refinement:

Theorem 4. *Let $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ be specifications with the same signature and let $\mathbf{A}$ and $\mathbf{B}$ be deterministic. If $\mathbf{A} \wedge \mathbf{B}$ is consistent then*

1. *$\rho(\mathbf{A} \wedge \mathbf{B}) \leq \mathbf{A}$ and $\rho(\mathbf{A} \wedge \mathbf{B}) \leq \mathbf{B}$,*
2. *$\mathbf{C} \leq \mathbf{A}$ and $\mathbf{C} \leq \mathbf{B}$ implies $\mathbf{C} \leq \rho(\mathbf{A} \wedge \mathbf{B})$,*
3. *$Impl(\rho(\mathbf{A} \wedge \mathbf{B})) = Impl(\mathbf{A}) \cap Impl(\mathbf{B})$.*

Quotient as the dual operator to structural composition. The quotient operator allows factoring out behaviors from larger specifications. Given two specifications $\mathbf{A}$ and $\mathbf{B}$ the quotient of $\mathbf{B}$ by $\mathbf{A}$, in the following denoted $\mathbf{B} \setminus\!\!\setminus \mathbf{A}$, is the most general specification that can be composed with $\mathbf{A}$ and still refines $\mathbf{B}$.

In the following, we assume for the signatures $Sig_{\mathbf{A}} = (\Sigma, V^L_{\mathbf{A}}, V^G_{\mathbf{A}})$ and $Sig_{\mathbf{B}} = (\Sigma, V^L_{\mathbf{B}}, V^G_{\mathbf{B}})$ that $V^L_{\mathbf{A}} \subseteq V^L_{\mathbf{B}}$. The signature of the quotient $\mathbf{B} \setminus\!\!\setminus \mathbf{A}$ is then $Sig_{\mathbf{B}\setminus\!\setminus\mathbf{A}} = (\Sigma, V^L_{\mathbf{B}\setminus\!\setminus\mathbf{A}}, V^G_{\mathbf{B}\setminus\!\setminus\mathbf{A}})$ with $V^L_{\mathbf{B}\setminus\!\setminus\mathbf{A}} = V^L_{\mathbf{B}} \setminus V^L_{\mathbf{A}}$ and $V^G_{\mathbf{B}\setminus\!\setminus\mathbf{A}} = V^G_{\mathbf{B}} \cup V^L_{\mathbf{A}}$. Note that, as said before, we restrict ourselves to the case where $V^L_{\mathbf{A}} \uplus V^G_{\mathbf{A}} = V^L_{\mathbf{B}} \uplus V^G_{\mathbf{B}}$.

It is unknown if in our general model of specifications a finite quotient exists. For specifications involving variables with finite domains only, a semantic quotient operation can be defined, which works on the (finite) semantics of $\mathbf{A}$ and $\mathbf{B}$. As already noticed in previous works, e.g. [17], non-determinism is problematic for quotienting, and thus specifications are assumed to be deterministic. In our case, even when assuming deterministic specifications, the non-determinism with respect to the next local data state is still there: thus the quotient $\mathbf{B} \setminus\!\!\setminus \mathbf{A}$, when performing a transition, does not know the next data state of $\mathbf{A}$. However, due to our semantics, in which transitions are guarded by uncontrolled states, the quotient can always observe the current data state of $\mathbf{A}$. This extension of the usual quotient can be shown that it satisfies the following soundness and maximality property: Given MSD $\mathbf{S}$ and $\mathbf{T}$ such that $\mathbf{S}$ is deterministic and $\mathbf{T} \setminus\!\!\setminus_{\mathrm{sem}} \mathbf{S}$ is consistent, and assume a semantic pruning operator ρ_{sem} which is the straightforward translation of pruning ρ to the semantic level. Then $\mathbf{X} \leq_{\mathrm{sem}} \rho_{\mathrm{sem}}(\mathbf{T} \setminus\!\!\setminus_{\mathrm{sem}} \mathbf{S})$ if and only if $\mathbf{S} \parallel_{\mathrm{sem}} \mathbf{X} \leq_{\mathrm{sem}} \mathbf{T}$ for any MSD $\mathbf{X}$.

Now our goal is to compute the quotient at the symbolic level of specifications. We do this for a restricted subclass of specifications in which each occurring transition predicate ψ is *separable*, meaning that ψ is equivalent to $^{\circ}\psi \wedge \psi^{\circ}$. Although this might seem as a serious restriction, we can often transform the transition systems with transition predicates of the form $(x)' = x + 1$ to a transition system with transition predicates which are separable and keep the same set of implementations. For instance, if we know that there are only finitely many possible values $v_1, \ldots, v_n$ for x in the current state, we can "unfold" the specification and replace the transition predicates $(x)' = x + 1$ by $(x)' = v_i$, for $1 \leq i \leq n$.

The symbolic quotient introduces two new locations, the universal state (univ) and an error state ($\bot$). In the universal state the quotient can show arbitrary behavior and is needed to obtain maximality, and the error state is a syntactically inconsistent state used to encode conflicting requirements. The state space of the quotient is given by $Loc_{\mathbf{B}} \times Loc_{\mathbf{A}} \times Pred(V^L_{\mathbf{A}})$, so every state stores not only the current location of $\mathbf{B}$ and $\mathbf{A}$ (like in [17]) but includes a predicate about the current possible data states of $\mathbf{A}$. For notational convenience, for $\varphi \in Pred(V_1 \uplus V_2)$ and $\varphi_1 \in Pred(V_1)$, we write $\varphi \mathbin{\backslash\!\!\backslash} \varphi_1$ for $(\forall V_1 . \varphi_1 \Rightarrow \varphi) \in Pred(V_2)$.

Definition 7. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two specifications such that* $V^L_{\mathbf{A}} \subseteq V^L_{\mathbf{B}}$. *The* quotient of $\mathbf{B}$ by $\mathbf{A}$ *is defined as the possibly syntactically inconsistent specification* $\mathbf{B} \mathbin{\backslash\!\!\backslash} \mathbf{A} = (Sig_{\mathbf{B}\mathbin{\backslash\!\!\backslash}\mathbf{A}}, (Loc_{\mathbf{B}} \times Loc_{\mathbf{A}} \times Pred(V^L_{\mathbf{A}})) \cup \{\mathsf{univ}, \bot\}, (\ell^0_{\mathbf{B}}, \ell^0_{\mathbf{A}}, \varphi^0_{\mathbf{A}}), \varphi^0_{\mathbf{B}} \mathbin{\backslash\!\!\backslash} \varphi^0_{\mathbf{A}}, E_\Diamond, E_\Box)$ *where the transition relations are given by, for all* $a \in \Sigma$ *and all* $\xi_{\mathbf{A}} \in Pred(V^L_{\mathbf{A}})$,

1. *if* $(\ell_{\mathbf{B}}, a, \psi_{\mathbf{B}}, \ell'_{\mathbf{B}}) \in E_{\Diamond,\mathbf{B}}$ *and* $(\ell_{\mathbf{A}}, a, \psi_{\mathbf{A}}, \ell'_{\mathbf{A}}) \in E_{\Diamond,\mathbf{A}}$, *then* $((\ell_{\mathbf{B}}, \ell_{\mathbf{A}}, \xi_{\mathbf{A}}), a, \xi_{\mathbf{A}} \wedge {}^\circ\psi_{\mathbf{B}} \wedge {}^\circ\psi_{\mathbf{A}} \wedge (\psi^\circ_{\mathbf{B}} \mathbin{\backslash\!\!\backslash} \psi^\circ_{\mathbf{A}}), (\ell'_{\mathbf{B}}, \ell'_{\mathbf{A}}, \psi^\circ_{\mathbf{A}})) \in E_\Diamond$,
2. *if* $(\ell_{\mathbf{B}}, a, \psi_{\mathbf{B}}, \ell'_{\mathbf{B}}) \in E_{\Box,\mathbf{B}}$ *and* $(\ell_{\mathbf{A}}, a, \psi_{\mathbf{A}}, \ell'_{\mathbf{A}}) \in E_{\Box,\mathbf{A}}$, *then* $((\ell_{\mathbf{B}}, \ell_{\mathbf{A}}, \xi_{\mathbf{A}}), a, \xi_{\mathbf{A}} \wedge {}^\circ\psi_{\mathbf{B}} \wedge {}^\circ\psi_{\mathbf{A}} \wedge (\psi^\circ_{\mathbf{B}} \mathbin{\backslash\!\!\backslash} \psi^\circ_{\mathbf{A}}), (\ell'_{\mathbf{B}}, \ell'_{\mathbf{A}}, \psi^\circ_{\mathbf{A}})) \in E_\Box$,
3. *if* $(\ell_{\mathbf{B}}, a, \psi_{\mathbf{B}}, \ell'_{\mathbf{B}}) \in E_{\Box,\mathbf{B}}$ *and* $(\ell_{\mathbf{A}}, a, \psi_{\mathbf{A}}, \ell'_{\mathbf{A}}) \in E_{\Box,\mathbf{A}}$, *then* $((\ell_{\mathbf{B}}, \ell_{\mathbf{A}}, \xi_{\mathbf{A}}), a, \xi_{\mathbf{A}} \wedge {}^\circ\psi_{\mathbf{B}} \wedge {}^\circ\psi_{\mathbf{A}} \wedge \neg(\psi^\circ_{\mathbf{B}} \mathbin{\backslash\!\!\backslash} \psi^\circ_{\mathbf{A}}), \bot) \in E_\Box$,
4. *if* $(\ell_{\mathbf{B}}, a, \psi_{\mathbf{B}}, \ell'_{\mathbf{B}}) \in E_{\Box,\mathbf{B}}$, *then* $((\ell_{\mathbf{B}}, \ell_{\mathbf{A}}, \xi_{\mathbf{A}}), a, \xi_{\mathbf{A}} \wedge {}^\circ\psi_{\mathbf{B}} \wedge \neg(\bigvee_{\psi_{\mathbf{A}} \in M} {}^\circ\psi_{\mathbf{A}}), \bot) \in E_\Box$ *where* $M = \bigcup_{\ell'_{\mathbf{A}} \in Loc_{\mathbf{A}}} Must^a_{\mathbf{A}}(\ell_{\mathbf{A}}, \ell'_{\mathbf{A}})$,
5. $((\ell_{\mathbf{B}}, \ell_{\mathbf{A}}, \xi_{\mathbf{A}}), a, \neg\xi_{\mathbf{A}}, \mathsf{univ}) \in E_\Diamond$,
6. $((\ell_{\mathbf{B}}, \ell_{\mathbf{A}}, \xi_{\mathbf{A}}), a, \xi_{\mathbf{A}} \wedge \neg(\bigvee_{\psi_{\mathbf{A}} \in M} {}^\circ\psi_{\mathbf{A}}), \mathsf{univ}) \in E_\Diamond$ *where* $M = \bigcup_{\ell'_{\mathbf{A}} \in Loc_{\mathbf{A}}} May^a_{\mathbf{A}}(\ell_{\mathbf{A}}, \ell'_{\mathbf{A}})$,
7. $(\mathsf{univ}, a, true, \mathsf{univ}) \in E_\Diamond$,
8. $(\bot, a, true, \bot) \in E_\Box$.

Rules (1) and (2) capture the cases when both $\mathbf{A}$ and $\mathbf{B}$ can perform a may- and must-transition, respectively. Rules (3) and (4) capture any inconsistencies which can arise if for a must-transition in $\mathbf{B}$ there is no way to obtain a must-transition by composition of the quotient with $\mathbf{A}$. In order to obtain maximality, we add a universal state univ in which the behavior of the quotient is not restricted (rules (5)–(7)). Finally, the rule (8) makes the error state syntactically inconsistent.

Since we only have finitely many transition predicates $\psi_{\mathbf{A}}$ in $\mathbf{A}$, and they are all separable, the set of locations $(Loc_{\mathbf{B}} \times Loc_{\mathbf{A}} \times (\{\psi^\circ_{\mathbf{A}} \mid \psi_{\mathbf{A}} \text{ occurring in } \mathbf{A}\} \cup \{\varphi^0_{\mathbf{A}}\})) \cup \{\mathsf{univ}, \bot\}$ of $\mathbf{B} \mathbin{\backslash\!\!\backslash} \mathbf{A}$ is also finite. Thus we can construct the symbolic quotient in a finite number of steps, starting in the initial state $(\ell^0_{\mathbf{B}}, \ell^0_{\mathbf{A}}, \varphi^0_{\mathbf{A}})$, and iteratively constructing the transitions. Soundness and maximality of the quotient follows from the following theorem.

Theorem 5. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be specifications such that* $V^L_{\mathbf{A}} \subseteq V^L_{\mathbf{B}}$, *all transition predicates of* $\mathbf{A}$ *and* $\mathbf{B}$ *are separable,* $\mathbf{A}$ *is deterministic and* $\mathbf{B} \mathbin{\backslash\!\!\backslash} \mathbf{A}$ *is consistent. Then for any specification* $\mathbf{C}$ *such that* $Sig_{\mathbf{C}} = Sig_{\mathbf{B}\mathbin{\backslash\!\!\backslash}\mathbf{A}}$, $\mathbf{C} \leq \rho(\mathbf{B} \mathbin{\backslash\!\!\backslash} \mathbf{A})$ *if and only if* $\mathbf{A} \parallel \mathbf{C} \leq \mathbf{B}$.

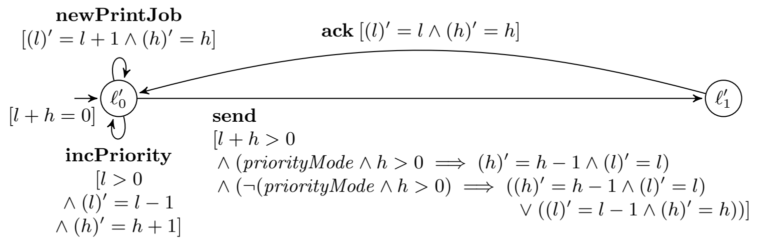
newPrintJob
$[(l)' = l + 1 \wedge (h)' = h]$
ack $[(l)' = l \wedge (h)' = h]$
ℓ'_0
ℓ'_1
$[l + h = 0]$
send
$[l + h > 0$
$\wedge\ (\mathit{priorityMode} \wedge h > 0 \implies (h)' = h - 1 \wedge (l)' = l)$
$\wedge\ (\neg(\mathit{priorityMode} \wedge h > 0) \implies ((h)' = h - 1 \wedge (l)' = l)$
$\vee\ ((l)' = l - 1 \wedge (h)' = h))]$
incPriority
$[l > 0$
$\wedge\ (l)' = l - 1$
$\wedge\ (h)' = h + 1]$

Fig. 5. Refined print server specification $\mathbf{Q}$.

4 Predicate Abstraction for Verification of Refinement

We now switch our focus to the problem of deciding whether a specification $\mathbf{A}$ refines another specification $\mathbf{B}$ (which reduces to checking $\langle \mathbf{A} \rangle_{\text{sem}} \leq_{\text{sem}} \langle \mathbf{B} \rangle_{\text{sem}}$). As soon as domains of variables are infinite, $\langle \mathbf{A} \rangle_{\text{sem}}$ and $\langle \mathbf{B} \rangle_{\text{sem}}$ may be MSD with infinitely many states and transitions. In this case, this problem is known to be undecidable in general. Thus we propose to resort to predicate abstraction techniques [18]. Given two specifications $\mathbf{A}$ and $\mathbf{B}$ we derive over- and under-approximations $\mathbf{A}^o$ and $\mathbf{B}^u$ which are guaranteed to be *finite* MSD. Then, we show that $\mathbf{A}^o \leq_{\text{sem}} \mathbf{B}^u$ implies $\mathbf{A} \leq \mathbf{B}$.

Example 5. Fig. 5 shows a print server specification $\mathbf{Q}$ which we will show is a refinement of the abstract specification $\mathbf{P}$ in Fig. 1. The behavior of the print server is now fixed for any number of print jobs. Moreover, the send transition has been refined such that depending on the priority mode (provided by the environment of the print server) a job with high priority (in case $\mathit{priorityMode}$ is true) or a job with low priority (otherwise) is chosen next.

Given a specification $\mathbf{A} = (\mathit{Sig}, \mathit{Loc}, \ell^0, \varphi^0, \longrightarrow_\Diamond, \longrightarrow_\Box)$ with $\mathit{Sig} = (\Sigma, V^L, V^G)$, we partition the local state space and the uncontrolled state space using finitely many predicates $\phi_1, \phi_2, \ldots, \phi_N \in \mathit{Pred}(V^L)$ and $\chi_1, \chi_2, \ldots, \chi_M \in \mathit{Pred}(V^G)$. We fix these predicates in the following to simplify the presentation. The signature of the abstraction is then given by $\mathit{Sig}_{\mathit{abstr}} = (\Sigma, V^L_{\mathit{abstr}}, V^G_{\mathit{abstr}})$, where $V^L_{\mathit{abstr}} = \{x_1, x_2, \ldots, x_N\}$ and $V^G_{\mathit{abstr}} = \{y_1, y_2, \ldots, y_M\}$. All variables x_i, y_j have Boolean domain. A variable x_i (y_j) encodes whether the predicate ϕ_i (χ_j) holds or not.

Any abstract state $\nu \in [\![V^L_{\mathit{abstr}}]\!]$ is a conjunction of predicates $\bigwedge_{i=1}^{N} \phi_i^{\nu(x_i)}$, where $\phi_i^{\nu(x_i)} = \phi_i$ if $\nu(x_i) = 1$, else $\phi_i^{\nu(x_i)} = \neg\phi_i$. Further, a set of abstract states $N \subseteq [\![V^L_{\mathit{abstr}}]\!]$ corresponds to $\bigvee_{\nu \in N} \nu$. Similarly for any $\omega \in [\![V^G_{\mathit{abstr}}]\!]$ and for $M \subseteq [\![V^G_{\mathit{abstr}}]\!]$.

The transition relation of the over-approximation expands the allowed behaviors and limits the required behaviors. Dually, the under-approximation will further restrict the allowed behavior and add more required transitions. In other words, over-approximation is an *existential* abstraction on may-transitions and *universal* abstraction on must-transitions; dually for the under-approximation.

Formally, the *over-approximation* $\mathbf{A}^o$ of $\mathbf{A}$ is defined by the finite TSD $(\mathit{Sig}_{\mathit{abstr}}, \mathit{Loc}, \ell^0, S^0_{\mathit{abstr}}, \longrightarrow_{\Diamond,\mathit{abstr}}, \longrightarrow_{\Box,\mathit{abstr}})$, where the initial abstract state contains all partitions overlapping with concrete initial states $S^0_{\mathit{abstr}} = \{\nu \in [\![V^L_{\mathit{abstr}}]\!] \mid \exists V^L . \nu \wedge \varphi^0\}$, and the abstract transition relations are derived as follows. For all $\ell, \ell' \in \mathit{Loc}$, $a \in \mathit{Act}$, $\nu, \dot{\nu} \in [\![V^L_{abstr}]\!]$, $\omega \in [\![V^G_{abstr}]\!]$,

i. If $\exists V.\exists (V^L)'.\nu \wedge \omega \wedge (\bigvee_{\psi \in May^a(\ell,\ell')} \psi) \wedge (\dot{\nu})'$, then $(\ell, \nu) \xrightarrow{\omega\, a}_{\Diamond, abstr} (\ell', \{\dot{\nu}\})$, so there is a may-transition between partitions in the abstraction if there was a may-transition between any states in these partitions in the concrete system.
ii. Whenever, for some $N \subseteq [\![V^L_{abstr}]\!]$, the predicate

$$\forall V.\nu \wedge \omega \Rightarrow \bigvee_{\psi \in Must^a(\ell,\ell')} {}^{\circ}\psi \wedge (\forall (V^L)'.\psi \Rightarrow (N)') \tag{2}$$

is true and N is minimal with respect to this property, then $(\ell, \nu) \xrightarrow{\omega\, a}_{\Box, abstr} (\ell', N)$.

For the *under-approximation* $\mathbf{B}^u$ of $\mathbf{B}$, we assume that every transition predicate ψ on a must-transition must be separable (see page 12). Moreover, in order to soundly capture must-transitions, we must be able to exactly describe the target set of (concrete) local states by a union of abstract states; so for any $(\ell, a, \psi, \ell') \in E_{\Box,\mathbf{B}}$, there exists a set $N \subseteq [\![V^L_{abstr}]\!]$ such that $\forall (V^L)'.\, \psi^{\circ} \Leftrightarrow (N)'$. The under-approximation $\mathbf{B}^u$ is the finite TSD $(\mathit{Sig}_{abstr}, \mathit{Loc}, \ell^0, S^0_{abstr}, \longrightarrow_{\Diamond, abstr}, \longrightarrow_{\Box, abstr})$, where $S^0_{abstr} = \{\nu \in [\![V^L_{abstr}]\!] \mid \forall V^L.\nu \Rightarrow \varphi^0\}$, and for all $\ell, \ell' \in \mathit{Loc}$, $a \in \mathit{Act}$, $\nu, \dot{\nu} \in [\![V^L_{abstr}]\!]$, $\omega \in [\![V^G_{abstr}]\!]$,

i. If $\forall V.\forall (V^L)'.\nu \wedge \omega \wedge (\dot{\nu})' \Rightarrow \bigvee_{\psi \in May^a(\ell,\ell')} \psi$ then $(\ell, \nu) \xrightarrow{\omega\, a}_{\Diamond, abstr} (\ell', \{\dot{\nu}\})$,
ii. For every $(\ell, a, \psi, \ell') \in E_{\Box,}$, if $\exists V.\nu \wedge \omega \wedge {}^{\circ}\psi$, then $(\ell, \nu) \xrightarrow{\omega\, a}_{\Box, abstr} (\ell', N)$ where $N \subseteq [\![V^L_{abstr}]\!]$ such that $\forall (V^L)'.\psi^{\circ} \Leftrightarrow (N)'$.

Correctness of the abstraction follows from the following theorem.

Theorem 6. *$\mathbf{A}^o \leq_{\text{sem}} \mathbf{B}^u$ implies $\mathbf{A} \leq \mathbf{B}$.*

Example 6. Fig. 6 and Fig. 7 are over- and under-approximations of $\mathbf{Q}$ and $\mathbf{P}$, respectively. The MSD represent abstractions w.r.t. the predicates $\phi_{0,0} \equiv h = l = 0$, $\phi_{0,1} \equiv l = 0 \wedge h = 1$, $\phi_{1,0} \equiv l = 1 \wedge h = 0$, and $\phi_{>1} \equiv h + l > 1$ for the controlled variables l and h, and $\omega_1 \equiv \mathit{priorityMode}$, $\omega_2 \equiv \neg \mathit{priorityMode}$ for the uncontrolled variable $\mathit{priorityMode}$. Note that all transition predicates in $\mathbf{P}$ are separable, and all possible (concrete) poststates can be precisely captured by the predicates $\phi_{0,0}, \phi_{0,1}, \phi_{1,0}, \phi_{>1}$. For better readability we have omitted most of the guards ω_1, ω_2, i.e. every transition without guard stands for two transitions with the same action, source and target state(s), and with ω_1 and ω_2 as guard, respectively. Moreover, the state $(\ell_3, \phi_{0,0} \vee \phi_{0,1} \vee \phi_{1,0} \vee \phi_{>1})$ is a simplified notation which represents all the states (ℓ_3, ϕ) with $\phi \in \{\phi_{0,0}, \phi_{0,1}, \phi_{1,0}, \phi_{>1}\}$ and all may-transitions leading to it lead to each of the states, and the may-loop stands for all the transitions between each of the states. Obviously, $\mathbf{Q}^o \leq_{\text{sem}} \mathbf{P}^u$, and from Thm. 6 it follows that $\mathbf{Q} \leq \mathbf{P}$.

Even though this abstraction technique requires separability of predicates, it is applicable to a larger set of specifications. Sometimes, as already described in the previous section, transitions with non-separable predicates can be replaced by finite sets of transitions to achieve separability, without changing the semantics of the specification. Automatic procedures for generation of predicates are subject of future work. Finally, our abstraction also supports compositional reasoning about parallel composition in the following sense:

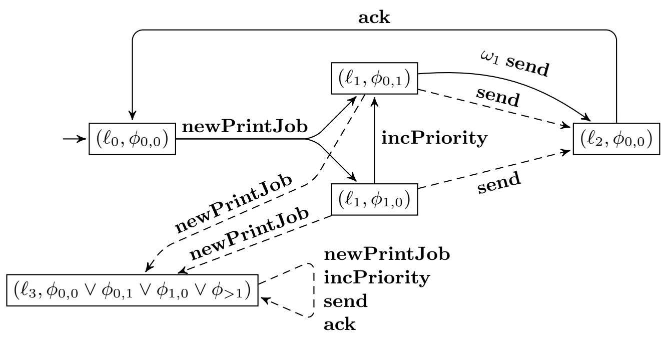

Fig. 6. Under-approximation $\mathbf{P}^u$.

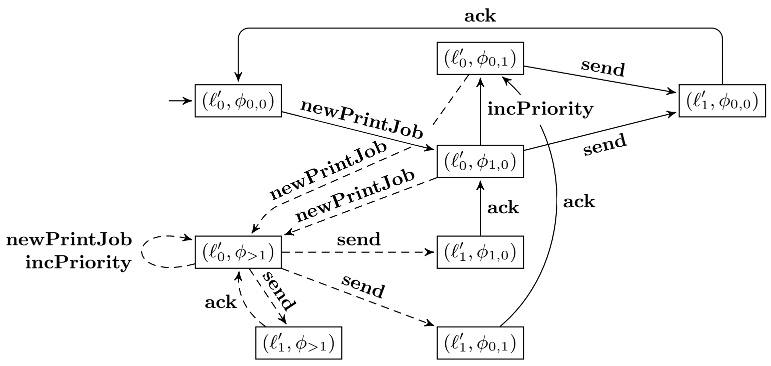

Fig. 7. Over-approximation $\mathbf{Q}^o$.

Theorem 7. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two composable specifications, and* $V^G_{\mathbf{A}\|\mathbf{B}} = (V^G_{\mathbf{A}} \cup V^G_{\mathbf{B}}) \setminus (V^L_{\mathbf{A}} \uplus V^L_{\mathbf{B}})$. *Let* $E_{\mathbf{A}} \subseteq \mathit{Pred}(V^L_{\mathbf{A}})$, $E_{\mathbf{A}} \subseteq \mathit{Pred}(V^L_{\mathbf{B}})$, *and* $F \subseteq \mathit{Pred}(V^G_{\mathbf{A}\|\mathbf{B}})$ *be sets of predicates partitioning the respective data states.*

$\mathbf{A}$ *is approximated w.r.t.* $E_{\mathbf{A}}$ *for* $V^L_{\mathbf{A}}$, *and* $E_{\mathbf{B}} \cup F$ *for* $V^G_{\mathbf{A}} = V^G_{\mathbf{A}\|\mathbf{B}} \uplus V^L_{\mathbf{B}}$ *and similarly,* $\mathbf{B}$ *is approximated w.r.t.* $E_{\mathbf{B}}$ *and* $E_{\mathbf{A}} \cup F$. *Finally,* $\mathbf{A} \parallel \mathbf{B}$ *is approximated w.r.t.* $E_{\mathbf{A}} \cup E_{\mathbf{B}}$ *for* $V^L_{\mathbf{A}\|\mathbf{B}} = V^L_{\mathbf{A}} \uplus V^L_{\mathbf{B}}$, *and* F *for* $V^G_{\mathbf{A}\|\mathbf{B}}$. *We assume that each predicate, in any abstraction of* $\mathbf{A}$, $\mathbf{B}$, *or* $\mathbf{A} \parallel \mathbf{B}$, *are encoded with the same variable.*

Then $(\mathbf{A} \parallel \mathbf{B})^o \leq_{\text{sem}} \mathbf{A}^o \parallel_{\text{sem}} \mathbf{B}^o$, *and* $\mathbf{A}^u \parallel_{\text{sem}} \mathbf{B}^u \leq_{\text{sem}} (\mathbf{A} \parallel \mathbf{B})^u$.

This result allows reusing abstractions of individual components in a continued development and verification process. For instance, if we want to verify $\mathbf{A} \parallel \mathbf{B} \leq \mathbf{C}$ then we can compute (or reuse) the less complex abstractions $\mathbf{A}^o$ and $\mathbf{B}^o$. Thm. 7 implies then that from $\mathbf{A}^o \parallel_{\text{sem}} \mathbf{B}^o \leq_{\text{sem}} \mathbf{C}^u$ we can infer $\mathbf{A} \parallel \mathbf{B} \leq \mathbf{C}$.

5 Related work

The main difference to related approaches based on modal process algebra taking data states into account, e.g. [19] is that they cannot naturally express logical and structural composition in the same formalism. A comparison between modal specifications and other theories such as interface automata [20] and process algebra [2] can be found in [3]. In [8], the authors introduced sociable interfaces, that is a model of I/O automata [21] equipped with a data and a game-based semantics. While their communication primitives are richer, sociable interfaces do not encompass any notion of logical composition and quotient, and their refinement is based on an alternating simulation.

Transition systems enriched with predicates are used, for instance, in the approach of [22, 23] where they use symbolic transition systems (STS), but STS do not support modalities and loose data specifications as they focus more on model checking than on the (top down) development of concurrent systems by refinement.

In [15] modal I/O automata has been extended by pre- and postconditions viewed as contracts, however, only semantics in terms of sets of implementations have been defined (implementations with only input actions correspond to our TSD). Modal refinement as defined in [15] is coarser than in this paper, and moreover, neither conjunction nor a quotient operation are defined.

6 Conclusion

We have proposed a specification theory for reasoning about components with rich data state. Our formalism, based on modal transition systems, supports: refinement checking, consistency checking with pruning of inconsistent states, structural and logical composition, and a quotient operator. We have defined symbolic representations of the operators and have shown that they are equivalent to the semantic definitions—this allows for automatic analysis of specifications. We have also presented a predicate abstraction technique for modal specifications with data. We believe that this work is a significant step towards practical use of specification theories based on modal transition systems. The ability to reason about data domains permits the modeling of industrial case studies.

In the future, we intend to develop larger case studies. Furthermore, we would like to extend the formalism with more complex communication patterns and to investigate in which cases we can still obtain all the operators on specifications, in particular the quotient operator. We are also planning to implement the theory in the MIO Workbench [24, 25, 26], a verification tool for modal input/output interfaces.

References

[1] Larsen, K.G.: Modal specifications. In Sifakis, J., ed.: Automatic Verification Methods for Finite State Systems. Volume 407 of Lecture Notes in Computer Science., Springer (1989)

[2] Milner, R.: A Calculus of Communicating Systems. Volume 92 of Lecture Notes in Computer Science. Springer (1980)

[3] Nyman, U.: Modal Transition Systems as the Basis for Interface Theories and Product Lines. PhD thesis, Department of Computer Science, Aalborg University (October 2008)

[4] Raclet, J.B., Badouel, E., Benveniste, A., Caillaud, B., Legay, A., Passerone, R.: Modal interfaces: unifying interface automata and modal specifications. In Chakraborty, S., Halbwachs, N., eds.: EMSOFT, ACM (2009)

[5] Abdulla, P.A., Bouajjani, A., d'Orso, J.: Monotonic and downward closed games. J. Log. Comput. **18**(1) (2008) 153–169

[6] Godefroid, P., Huth, M., Jagadeesan, R.: Abstraction-based model checking using modal transition systems. In: CONCUR. (2001) 426–440

[7] Abdulla, P.A., Cerans, K., Jonsson, B., Tsay, Y.K.: Algorithmic analysis of programs with well quasi-ordered domains. Inf. Comput. **160**(1-2) (2000) 109–127

[8] de Alfaro, L., da Silva, L.D., Faella, M., Legay, A., Roy, P., Sorea, M.: Sociable interfaces. In Gramlich, B., ed.: FroCos. Volume 3717 of Lecture Notes in Computer Science., Springer (2005) 81–105

[9] Larsen, K.G., Xinxin, L.: Equation solving using modal transition systems. In: LICS, IEEE Computer Society (1990) 108–117

[10] Larsen, K.G., Thomsen, B.: A modal process logic. In: LICS, IEEE Computer Society (1988) 203–210

[11] Antonik, A., Huth, M., Larsen, K.G., Nyman, U., Wasowski, A.: Complexity of decision problems for mixed and modal specifications. In: FoSSaCS 2008. Volume 4962 of Lecture Notes in Computer Science., Springer (2008) 112–126

[12] Larsen, K.G., Nyman, U., Wąsowski, A.: On modal refinement and consistency. In Caires, L., Vasconcelos, V.T., eds.: CONCUR. Volume 4703 of Lecture Notes in Computer Science., Springer (2007) 105–119

[13] Benes, N., Kretínský, J., Larsen, K.G., Srba, J.: On determinism in modal transition systems. Theor. Comput. Sci. **410**(41) (2009) 4026–4043

[14] de Alfaro, L., Henzinger, T.A.: Interface theories for component-based design. In: Embedded Software, First International Workshop, EMSOFT 2001, Tahoe City, CA, USA, October, 8-10, 2001, Proceedings. Volume 2211 of Lecture Notes in Computer Science., Springer (2001) 148–165

[15] Bauer, S.S., Hennicker, R., Wirsing, M.: Interface theories for concurrency and data. Theor. Comput. Sci. (2011) To appear.

[16] Larsen, K.G., Nyman, U., Wąsowski, A.: Modal I/O automata for interface and product line theories. In Nicola, R.D., ed.: ESOP. Volume 4421 of Lecture Notes in Computer Science., Springer (2007) 64–79

[17] Raclet, J.B.: Residual for component specifications. Electr. Notes Theor. Comput. Sci. **215** (2008) 93–110

[18] Graf, S., Saïdi, H.: Construction of abstract state graphs with pvs. In Grumberg, O., ed.: CAV. Volume 1254 of Lecture Notes in Computer Science., Springer (1997) 72–83

[19] van de Pol, J., Espada, M.V.: Modal Abstractions in μCRL. In Rattray, C., Maharaj, S., Shankland, C., eds.: AMAST. Volume 3116 of Lecture Notes in Computer Science., Springer (2004) 409–425

[20] de Alfaro, L., Henzinger, T.A.: Interface automata. SIGSOFT Softw. Eng. Notes **26** (September 2001) 109–120

[21] Lynch, N., Tuttle, M.R.: An introduction to Input/Output automata. CWI-quarterly **2**(3) (1989)

[22] Fernandes, F., Royer, J.C.: The STSLib project: Towards a formal component model based on STS. Electr. Notes Theor. Comput. Sci. **215** (2008) 131–149

[23] Barros, T., Ameur-Boulifa, R., Cansado, A., Henrio, L., Madelaine, E.: Behavioural models for distributed fractal components. Annales des Télécommunications **64**(1-2) (2009) 25–43

[24] Bauer, S.S., Mayer, P., Schroeder, A., Hennicker, R.: On weak modal compatibility, refinement, and the mio workbench. In Esparza, J., Majumdar, R., eds.: TACAS. Volume 6015 of Lecture Notes in Computer Science., Springer (2010) 175–189

[25] Bauer, S.S., Mayer, P., Legay, A.: MIO Workbench: A Tool for Compositional Design with Modal Input/Output Interfaces. In: Automated Technology for Verification and Analysis (ATVA 2011). (2011) Accepted for publication.

[26] MIO Workbench: http://www.miowb.net/