Compositionality of Hennessy-Milner Logic by Structural Operational Semantics

Wan Fokkink[1,4], Rob van Glabbeek[1,2,3], Paulien de Wind[4]

1: CWI

Department of Software Engineering, PO Box 94079,

1090 GB Amsterdam, The Netherlands

2: National ICT Australia

Locked Bag 6016, University of New South Wales

Sydney 1466, Australia

3: University of New South Wales

School of Computer Science and Engineering,

Sydney 2052, Australia

4: Vrije Universiteit Amsterdam

Department of Theoretical Computer Science,

De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

email: wanf@cs.vu.nl, rvg@cs.stanford.edu, pdwind@cs.vu.nl

Abstract

This paper presents a method for the decomposition of HML formulas. It can be used to decide whether a process algebra term satisfies a HML formula, by checking whether subterms satisfy certain formulas, obtained by decomposing the original formula. The method uses the structural operational semantics of the process algebra. The main contribution of this paper is the extension of an earlier decomposition method for the De Simone format from the PhD thesis of Kim G. Larsen in 1986, to more general formats.

1 Introduction

In the past two decades, compositional methods have been developed for checking the validity of assertions in modal logics, used to describe the behaviour of

processes. This means that the truth of an assertion for a composition of processes can be deduced from the truth of certain assertions for its components. Most research papers in this area focus on a particular process algebra.

Barringer, Kuiper & Pnueli presented in [4] (a preliminary version of) a compositional proof system for concurrent programs, which is based on a rich temporal logic, including operators from process logic [15] and LTL [26]. For modelling concurrent programs they defined a language including assignment, conditional and while statements. Parallel components interact via shared variables.

In [29], Stirling developed modal proof systems for subsets of CCS [22] (with and without silent actions) including only sequential and alternative composition, to decide the validity of formulas from Hennessy-Milner Logic (HML) [16]. In [31, 30], Stirling extended the results from [29], creating proof systems for subsets of CCS and SCCS [24] including asynchronous and synchronous parallelism and infinite behaviour, using ideas from [4]. In [32], Stirling generalised the proposals from [31, 30] in order to cope with the restriction operator.

Winskel gave in [33] a method to decompose formulas with respect to each operation in SCCS. The language of assertions is HML with infinite conjunction and disjunction. This decomposition provides the foundations of Winskel's proof system for SCCS with modal assertions. In [34], [3] and [2] processes are described by specification languages inspired by CCS and CSP [8]. The articles describe compositional methods for deciding whether processes satisfy assertions from a modal μ-calculus [18].

Larsen developed in [20] a more general compositional method for deciding whether a process satisfies a certain property. Unlike the aforementioned methods, this method is not oriented towards a particular process algebra, but it is based on structural operational semantics [25], which provides process algebras and specification languages with an interpretation. A transition system specification, consisting of an algebraic signature and a set of transition rules of the form $\frac{\text{premises}}{\text{conclusion}}$, generates a transition relation between the closed terms over the signature. An example of a transition rule, for alternative composition, is

$$\frac{x_1 \xrightarrow{a} y}{x_1 + x_2 \xrightarrow{a} y}$$

meaning for states t_1, t_2 and u that if state t_1 can evolve into state u by the execution of action a, then so can state $t_1 + t_2$. Larsen showed how to decompose HML formulas with respect to a transition system specification in the De Simone format [27]. This format was originally put forward to guarantee that the bisimulation equivalence associated with a transition system specification is a congruence, meaning that bisimulation equivalence is preserved by all functions in the signature. In [21], Larsen and Liu extended this decomposition method to HML with recursion (which is equivalent to the modal μ-calculus).

Since modal proof systems for specific process algebras are tailor-made, they may be more concise than the ones generated by the general decomposition method of Larsen (e.g., [31, 30, 32]). However, in some cases the general de-

composition method does produce modal proof systems that are similar in spirit to those in the literature (e.g., [29, 33]).

For systems consisting of parallel compositions of interacting components and specification logics based on the modal μ-calculus, the efficacy of Larsen's compositional approach was demonstrated by Andersen in [1]. Laroussinie & Larsen applied this approach in [19] to real-time systems modelled as networks of timed automata.

Bloom, Fokkink & van Glabbeek presented in [5] a method for decomposing formulas from a fragment of HML with infinite conjunction, with respect to terms from any process algebra that has a structural operational semantics in ready simulation format, which is the ntyft/ntyxt format [13] without lookahead. A rule in ntyft/ntyxt format may contain negative premises, the left-hand side of the conclusion contains at most one function symbol and no multiple occurrences of variables, and the right-hand sides of positive premises are variables that are all distinct and do not occur in the left-hand side of the conclusion. Such a rule has no lookahead if variables in the right-hand sides of premises do not occur in the left-hand sides of premises. This format is a generalisation of the De Simone format, and still guarantees that bisimulation equivalence is a congruence. The decomposition method is not presented in its own right, but is used in the derivation of congruence formats for a range of behavioural equivalences from van Glabbeek [11].

In the current paper, the decomposition method from [5] is extended to full HML with infinite conjunction, for process algebras with a structural operational semantics in ntyft/ntyxt format without lookahead or in tyft/tyxt format [14]. The latter format is the same as the ntyft/ntyxt format, but disallows negative premises. The rules in this format may contain lookahead, i.e. there may be a chain of premises such that the right-hand side of each premise occurs in the left-hand side of the next premise; if this chain is finite, we speak of bounded lookahead. We show that if a rule has unbounded lookahead, it can be replaced by a rule with bounded lookahead. This is needed because no HML formula can capture unbounded lookahead.

In [28], Simpson presented a proof system for establishing the validity of HML formulas for processes in arbitrary languages with a structural operational semantics in GSOS format [6], using a mixture of compositional and structural styles of proof. His method differs from ours, and from the work referenced above, in that the validity of an assertion for a process term may be inferred from statements about the dynamic behaviour of that process, whereas we only allow modal assertions for subterms.

An earlier version of this paper, which featured only a compositionality result for ntyft/ntyxt format without lookahead, appeared as [10].

The structure of the paper is as follows. Section 2 contains the preliminaries on modal logic and structural operational semantics. In Section 3 we present the decomposition method for HML formulas. Finally, in Section 4 we show as an application of this method that the congruence theorem for the tyft/tyxt format from [14, 9] is an immediate corollary of our decomposition result.

2 Preliminaries

In this section we give the basic notions of Hennessy-Milner Logic and structural operational semantics that are needed to define our decomposition method.

definition 2.1 (LTS, transition relation) *A* labelled transition system (LTS) *is a pair* $(\mathbb{P}, \rightarrow)$ *with* $\mathbb{P}$ *a set of* processes *and* $\rightarrow \subseteq \mathbb{P} \times A \times \mathbb{P}$ *for* A *a set of* actions. *The relation* $\rightarrow$ *is called the* transition relation *and its elements are called* transitions. *We write* $p \xrightarrow{a} q$ *for* $(p, a, q) \in \rightarrow$ *and* $p \xrightarrow{a}\!\!\!\!\!/\;\;$ *for* $\neg\exists q \in \mathbb{P} : p \xrightarrow{a} q$.

The elements of $\mathbb{P}$ represent the processes we are interested in, and $p \xrightarrow{a} q$ means that process p can evolve into process q while performing the action a.

2.1 Hennessy-Milner Logic

Modal logic aims at formulating properties of processes, and to identify processes that satisfy the same properties. In [16], Hennessy & Milner defined a modal language, often called Hennessy-Milner Logic (HML), which characterises the bisimulation equivalence relation on processes, assuming that the transition relation is *image finite*, i.e. each process has finitely many outgoing a-transitions for each $a \in A$. This assumption can be discarded if infinite conjunctions are allowed; see [23, 17].

definition 2.2 (Hennessy-Milner Logic) *Assume an action set* A. *The set* $\mathbb{O}$ *of* potential observations *or* modal formulas *is recursively defined by*

$$\varphi ::= \bigwedge_{i \in I} \varphi_i \mid \langle a \rangle \varphi \mid \neg\varphi$$

with $a \in A$ *and* I *some index set.*

definition 2.3 (satisfaction relation) *Let* $(\mathbb{P}, \rightarrow)$ *be an LTS. The* satisfaction relation $\models \subseteq \mathbb{P} \times \mathbb{O}$ *is defined as follows, with* $p \in \mathbb{P}$*:*

$$\begin{array}{ll} p \models \bigwedge_{i \in I} \varphi_i & \text{iff } p \models \varphi_i \text{ for all } i \in I \\ p \models \langle a \rangle \varphi & \text{iff there is a } q \in \mathbb{P} \text{ such that } p \xrightarrow{a} q \text{ and } q \models \varphi \\ p \models \neg\varphi & \text{iff } p \not\models \varphi \end{array}$$

We will use the binary conjunction $\varphi_1 \wedge \varphi_2$ as an abbreviation of $\bigwedge_{i \in \{1,2\}} \varphi_i$, whereas $\top$ is an abbreviation for the empty conjunction. We identify formulas that are logically equivalent using the laws $\top \wedge \varphi \cong \varphi$, $\bigwedge_{i \in I}(\bigwedge_{j \in J_i} \varphi_j) \cong \bigwedge_{i \in I,\, j \in J_i} \varphi_j$ and $\neg\neg\varphi \cong \varphi$. This is justified because $\varphi \cong \psi$ implies $p \models \varphi \Leftrightarrow p \models \psi$.

definition 2.4 (bisimulation) *Let* $(\mathbb{P}, \rightarrow)$ *be an LTS. A binary relation* $\mathcal{B} \subseteq \mathbb{P} \times \mathbb{P}$ *is a* bisimulation *if it is symmetric and* $p\mathcal{B}q$ *implies: if* $p \xrightarrow{a} p'$ *then there is a* q' *such that* $q \xrightarrow{a} q'$ *and* $p'\mathcal{B}q'$. *For* $p, q \in \mathbb{P}$ *we write* $p \leftrightarrows q$ *if there is a bisimulation* $\mathcal{B}$ *such that* $p\mathcal{B}q$.

Proposition 2.5 ([23, 17]) *Let* $(\mathbb{P}, \rightarrow)$ *be an LTS. Then* $p \leftrightarrows q$ *if and only iff* $p \models \varphi \Leftrightarrow q \models \varphi$ *for all* $\varphi \in \mathbb{O}$.

2.2 Structural Operational Semantics

Structural operational semantics [25] provides a framework to give an operational semantics to programming and specification languages. In particular, because of its intuitive appeal and flexibility, structural operational semantics has found considerable application in the study of the semantics of concurrent processes.

Let V be a set of variables. If S is any syntactic object, $var(S)$ denotes the set of variables that occur in S. A syntactic object S is called *closed* if it does not contain any variables from V, i.e. if $var(S) = \emptyset$.

definition 2.6 (signature) *A* signature *is a collection* Σ *of* function symbols $f \notin V$*, equipped with a function* $ar : \Sigma \to \mathbb{N}$*. The set* $\mathbb{T}(\Sigma)$ *of* terms *over a signature* Σ *is defined recursively by:*

- $V \subseteq \mathbb{T}(\Sigma)$*,*
- *if* $f \in \Sigma$ *and* $t_1, \ldots, t_{ar(f)} \in \mathbb{T}(\Sigma)$*, then* $f(t_1, \ldots, t_{ar(f)}) \in \mathbb{T}(\Sigma)$*.*

A term $c()$ *is abbreviated as* c*.* $T(\Sigma)$ *is the set of closed terms over* Σ*. A* Σ-substitution σ *is a partial function from* V *to* $\mathbb{T}(\Sigma)$*. The domain of* σ *is denoted by* $dom(\sigma)$*. If* σ *is a* Σ*-substitution and* S *is any syntactic object, then* $\sigma(S)$ *denotes the object obtained from* S *by replacing, for* x *in the domain of* σ*, every occurrence of* x *in* S *by* $\sigma(x)$*. In that case* $\sigma(S)$ *is called a* substitution instance *of* S*. A* Σ*-substitution* σ *is* closed *if* $\sigma(x) \in T(\Sigma)$ *for all* $x \in V$*.*

In the remainder, let Σ denote a signature and A a set of actions.

definition 2.7 (literal) *A* positive Σ-literal *is an expression* $t \xrightarrow{a} t'$ *and a* negative Σ-literal *an expression* $t \xrightarrow{a}\!\!\!\!\!\!/\;\;$ *with* $t, t' \in \mathbb{T}(\Sigma)$ *and* $a \in A$*. For* $t, t' \in \mathbb{T}(\Sigma)$ *and* $a \in A$*, the literals* $t \xrightarrow{a} t'$ *and* $t \xrightarrow{a}\!\!\!\!\!\!/\;\;$ *are said to* deny *each other.*

definition 2.8 (transition rule) *A* transition rule *over* Σ *is an expression of the form* $\frac{H}{\alpha}$ *with* H *a set of* Σ*-literals (the* premises *of the the rule) and* α *a positive* Σ*-literal (the* conclusion*). With* $lhs(H)$ *and* $rhs(H)$ *we denote the sets of left- and right-hand sides of the premises in* H*, respectively. The left- and right-hand side of* α *are called the* source *and the* target *of the rule, respectively. A rule* $\frac{H}{\alpha}$ *with* $H = \emptyset$ *is also written* α*.*

definition 2.9 (transition system specification) *A* transition system specification (TSS) *is a pair* (Σ, R) *with* R *a collection of transition rules over* Σ*.*

The purpose of a TSS (Σ, R) is to specify an LTS of the form $(T(\Sigma), \rightarrow)$ with as processes the closed terms over Σ and as transition relation a set of closed positive literals $\rightarrow \subseteq T(\Sigma) \times A \times T(\Sigma)$. Hence we refer to a closed positive literal as a *transition*.

definition 2.10 (proof) *Let* $P = (\Sigma, R)$ *be a TSS. An* irredundant proof *from* P *of a transition rule* $\frac{H}{\alpha}$ *is a well-founded, upwardly branching tree whose nodes are labelled by* Σ*-literals, and some of the leaves are marked "hypothesis", such that:*

- *the root is labelled by* α,
- H *is the set of labels of the hypotheses, and*
- *if* β *is the label of a node* q *which is not a hypothesis and* K *is the set of labels of the nodes directly above* q*, then* $\frac{K}{\beta}$ *is a substitution instance of a transition rule in* R.

A proof *from* P *of* $\frac{K}{\alpha}$ *is an irredundant proof from* P *of* $\frac{H}{\alpha}$ *with* $H \subseteq K$. *If an (irredundant) proof from* P *of* $\frac{K}{\alpha}$ *exists, then* $\frac{K}{\alpha}$ *is* (irredundantly) provable *from* P*, notation* $P \vdash \frac{K}{\alpha}$ *(resp.* $P \vdash_{\mathrm{irr}} \frac{K}{\alpha}$*).*

The proof of $\frac{H}{\alpha}$ is called irredundant because H must equal (instead of include) the set of labels of the hypotheses. The main advantage of irredundant proofs is that derived rules may inherit certain syntactic structure from the transition rules in the TSS from which they are derived; in standard proofs this syntactic structure is usually lost, because arbitrary literals can be added as premises of derived rules. In the current paper, irredundancy of proofs is not of immediate importance, but it can be essential in applications of our decomposition method for modal formulas. For instance, in [5], irredundancy is essential to guarantee that the syntactic restrictions of congruence formats for TSSs are also satisfied by the rules derived from those TSSs.

A TSS with only positive premises specifies a transition relation in a straightforward way as the set of all provable transitions. But it is much less trivial to associate a transition relation to a TSS with negative premises. Several solutions are proposed by Groote in [13], Bol & Groote in [7], and van Glabbeek in [12]. From the last we adopt the notion of a well-supported proof and a complete TSS.

definition 2.11 (well-supported proof) *Let* $P = (\Sigma, R)$ *be a TSS. A* well-supported proof *from* P *of a closed literal* α *is a well-founded, upwardly branching tree whose nodes are labelled by closed* Σ*-literals, such that:*

- *the root is labelled by* α*, and*
- *if* β *is the label of a node* q *and* K *is the set of labels of the nodes directly above* q*, then*
  1. *either* $\frac{K}{\beta}$ *is a closed substitution instance of a transition rule in* R
  2. *or* β *is negative and for every set* N *of closed negative literals such that* $P \vdash \frac{N}{\gamma}$ *for* γ *a closed literal denying* β*, a literal in* K *denies one in* N.

We say α *is ws-*provable *from* P*, notation* $P \vdash_{ws} \alpha$*, if a well-supported proof from* P *of* α *exists.*

In [12] it was noted that $\vdash_{ws}$ is *consistent*, in the sense that no TSS admits well-supported proofs of two literals that deny each other.

definition 2.12 (completeness) *A TSS P is* complete *if for any closed literal $p \stackrel{a}{\not\longrightarrow}$ either $P \vdash_{ws} p \stackrel{a}{\longrightarrow} p'$ for some closed term p' or $P \vdash_{ws} p \stackrel{a}{\not\longrightarrow}$.*

Now a TSS specifies a transition relation if and only if it is complete. The specified transition relation is then the set of all *ws*-provable transitions. For positive TSSs, the set of $\vdash_{ws}$-provable closed positive literals and the set of $\vdash$-provable closed literals are the same.

Example 2.13 *Let $A = \{a, b\}$ and $P = (\Sigma, R)$, where Σ consists of the constant c and the unary function symbol f, and R is:*

$$\frac{x \stackrel{a}{\not\longrightarrow}}{f(x) \stackrel{b}{\longrightarrow} c} \qquad \frac{c \stackrel{a}{\not\longrightarrow}}{c \stackrel{a}{\longrightarrow} c}$$

There are well-supported proofs for the closed negative literals $f^{n+1}(c) \stackrel{a}{\not\longrightarrow}$ for $n \geq 0$ and $c \stackrel{b}{\not\longrightarrow}$. In each case, the proof consists of a single node, labelled by the literal itself. Furthermore, there are well-supported proofs for the closed positive literals $f^{n+2}(c) \stackrel{b}{\longrightarrow} c$ for $n \geq 0$, where the proof consists of a tree of two nodes: the root has as label the literal itself, and the node above it has label $f^{n+1}(c) \stackrel{a}{\not\longrightarrow}$. The TSS P is not complete, since there is no well-supported proof for $c \stackrel{a}{\not\longrightarrow}$ or $c \stackrel{a}{\longrightarrow} p$ for $p \in T(\Sigma)$; moreover, there is no well-supported proof for $f(c) \stackrel{b}{\not\longrightarrow}$ or $f(c) \stackrel{b}{\longrightarrow} p$ for $p \in T(\Sigma)$.

2.3 Formats

We recall the tyft/tyxt format [14] and the ready simulation format [5]. Both formats are restrictions of the ntyft/ntyxt format [13], which was introduced as the most liberal format known to guarantee congruence properties for bisimulation equivalence.

definition 2.14 (ntytt) *An* ntytt rule *is a transition rule in which the right-hand sides of positive premises are variables that are all distinct, and that do not occur in the source. An ntytt rule is an* ntyxt rule *if its source is a variable, and an* ntyft rule *if its source contains exactly one function symbol and no multiple occurrences of variables. An ntytt rule is an* nxytt rule *if the left-hand sides of its premises are variables. A* tytt, tyxt, tyft *or* xytt rule *is an ntytt, ntyxt, ntyft or nxytt rule, respectively, without negative premises. A* xyft rule *is a tyft rule that is also an xytt rule.*

Given a premise $t \stackrel{a}{\longrightarrow} y$ of an ntytt rule, there is a dependency between each variable in t and y. This is captured in the dependency graph of the rule. We will strive for rules in which there is neither infinite forward dependency between variables (bounded lookahead), nor infinite backward dependency between variables (well-foundedness).

definition 2.15 (lookahead, well-foundedness) *Assume a set of positive* Σ*-literals* $\{t_i \xrightarrow{a_i} t'_i \mid i \in I\}$*. Its* dependency graph *is a directed graph, with the collection of variables* V *as vertices, and with as edges*

$$\{\langle x, y\rangle \mid x \text{ and } y \text{ occur in } t_i \text{ and } t'_i \text{ respectively, for some } i \in I\}.$$

The dependency graph of a transition rule is the dependency graph of its positive premises. A transition rule is well-founded *if any backward chain of edges in its dependency graph is finite. A transition rule has* lookahead *if there is a variable in the right-hand side of a premise that also occurs in the left-hand side of a premise; the lookahead is* bounded *if any forward chain of edges in the dependency graph is finite.*

definition 2.16 (pure) *A variable in a transition rule is* free *if it occurs neither in the source nor in right-hand sides of positive premises of this rule. A rule is* pure *if it is well-founded and does not contain free variables.*

definition 2.17 (depth) *If* r *is a pure tytt rule of the form* $\frac{\{v_k \xrightarrow{c_k} y_k \mid k \in K\}}{t \xrightarrow{a} u}$*, then the* depth *of terms* v *with* $var(v) \subseteq var(r)$ *is defined by:*

$$\begin{array}{ll} depth_r(x) = 0 & \text{for } x \in var(t) \\ depth_r(y_k) = depth_r(v_k) + 1 & \text{for } k \in K \\ depth_r(v) = max\{depth_r(z) \mid z \in var(v)\} & \text{if } var(v) \subseteq var(r) \end{array}$$

This is well-defined since the set of premises is well-founded. When clear from the context, the subscript r *will be omitted. For* $d \geq 0$ *we write* K_d *for* $\{k \in K \mid depth(v_k) = d\}$*.*

Each combination of syntactic restrictions on transition rules induces a corresponding syntactic format of the same name for TSSs. For example, a TSS is in *tyft/tyxt format* if and only if it contains only tyft and tyxt rules.

definition 2.18 (ready simulation format) *A TSS is in* ready simulation format *if it contains only ntyft and ntyxt rules without lookahead.*

3 Decomposing HML Formulas

In this section it is shown how one can decompose HML formulas with respect to process terms. Section 3.1 explains what this means. The TSS defining the transition relation on these terms should be in ready simulation format or in tyft/tyxt format, i.e. in ntyft/ntyxt format and either without lookahead or without negative premises.

The decomposition method uses a collection of rules extracted from this TSS, called *ruloids*. These rules have the property that there is a well-supported proof of a transition $p \xrightarrow{a} q$, with p a closed substitution instance of a term t, if and only if there exists a proof that uses at the root a ruloid with source t (Theorems 3.2 and 3.9). We require our ruloids to be pure nxytt rules with

bounded lookahead. Moreover, when dealing with a TSS without lookahead, the ruloids are required to be without lookahead as well, and when dealing with a TSS without negative premises, the ruloids likewise have to be xytt rules.

The construction of the ruloids for a TSS in ready simulation format stems from Bloom, Fokkink & van Glabbeek [5] and is briefly recalled in Section 3.2. In Section 3.3 we construct ruloids for TSSs in tyft/tyxt format, applying the transformation from tyft/tyxt to pure xyft format from Fokkink & van Glabbeek [9]. An essential new step in the construction is the replacement of any pure nxytt rule by an equivalent rule with bounded lookahead.

The decomposition method is given in Section 3.4, together with the proof that a term t under closed substitution σ satisfies a formula if and only if the variables in t under substitution σ satisfy the formulas given by the decomposition method.

Section 3.6 contains two toy examples to illustrate the method. Section 3.7 features two counterexamples; one to underline the importance that TSSs are complete, and one to show why our two decomposition results for ready simulation and tyft/tyxt format cannot be combined in a straightforward fashion.

3.1 Some Intuition

To explain our decomposition method, we start with an example.

Example 3.1 *Consider the TSS with rules*

$$a.x \xrightarrow{a} x \qquad \frac{x \xrightarrow{a} x'}{x\|y \xrightarrow{a} x'\|y} \qquad \frac{y \xrightarrow{a} y'}{x\|y \xrightarrow{a} x\|y'}$$

for $a \in A$. The following rules describe all circumstances under which a process $x\|y$ satisfies the modal formula $\langle a\rangle\langle b\rangle\top$:

$$\frac{x \models \langle a\rangle\langle b\rangle\top}{x\|y \models \langle a\rangle\langle b\rangle\top} \qquad \frac{x \models \langle a\rangle\top \quad y \models \langle b\rangle\top}{x\|y \models \langle a\rangle\langle b\rangle\top} \qquad \frac{x \models \langle b\rangle\top \quad y \models \langle a\rangle\top}{x\|y \models \langle a\rangle\langle b\rangle\top} \qquad \frac{y \models \langle a\rangle\langle b\rangle\top}{x\|y \models \langle a\rangle\langle b\rangle\top}$$

In this paper we show how to produce, for any modal formula φ, a complete set of rules for deriving $x\|y \models \varphi$, whose premises have the form $x \models \psi$ or $y \models \psi$. As the number of rules we need grows dramatically in the size of φ, we cannot simply list these rules exhaustively. Instead we generate them with induction on the structure of φ. Let us start with proposing some notation. The set of rules above, all of which share the conclusion $x\|y \models \langle a\rangle\langle b\rangle\top$, is completely determined by their sets of premises. Each set of premises can be given as a partial function ψ that associates modal formulas $\psi(x)$ to variables x. The second rule, for instance, is given by the partial function ψ_2, defined by $\psi_2(x) = \langle a\rangle\top$ and $\psi_2(y) = \langle b\rangle\top$. For convenience, we extend these partial functions to total ones—in the first rule above by defining $\psi_1(y) = \top$. This is justified because the premise $y \models \top$ is vacuously true for any process y. The set of the four mappings ψ_1–ψ_4 characterising the four rules displayed above is called the modal decomposition *of the formula $\langle a\rangle\langle b\rangle\top$ with respect to the term $x\|y$ in the TSS above. This model decomposition is denoted $(x\|y)^{-1}(\langle a\rangle\langle b\rangle\top)$.*

In general, for a given term t and a modal formula φ, the modal decomposition $t^{-1}(\varphi)$ of φ w.r.t. t is a set of requirements ψ_i, one of which must be satisfied in order for t to satisfy φ. Each requirement ψ_i is given as a mapping from variables to modal formulas; it holds when for all $x \in var(t)$ the process represented by x satisfies $\psi_i(x)$.

The main contribution of this paper is the definition of $t^{-1}(\varphi)$ for any HML formula φ and any term t, relative to a complete TSS in ready simulation format or in tyft/tyxt format (Definition 3.10), together with the theorem that states that this definition is correct (Theorem 3.11):

> *For any closed substitution σ, the process $\sigma(t)$ satisfies φ exactly when there is a mapping ψ in the modal decomposition $t^{-1}(\varphi)$ such that $\sigma(x)$ satisfies $\psi(x)$ for all variables x in t.*

This model decomposition result can be *used* to infer the validity of a statement $t \models \varphi$ by structural induction on t. However, the result is *obtained* by structural induction on φ.

3.2 Ruloids for the Ready Simulation Format

In the decomposition of modal formulas, we need an important result from [5], where for any TSS P in ready simulation format a collection of pure nxytt rules, called *P-ruloids*, is constructed. We explain this construction on a rather superficial level; the precise transformation can be found in [5].

First P is converted to a TSS in pure ntyft format without lookahead. In this conversion from [14], free variables in a rule are replaced by closed terms, and if the source is of the form x then this variable is replaced by a term $f(x_1, \ldots, x_n)$ for each $f \in \Sigma$. Next, using a construction from [9], left-hand sides of positive premises are reduced to variables. Roughly the idea is, given a premise $f(t_1, \ldots, t_n) \xrightarrow{\alpha} y$ in a rule r, and a rule $\frac{H}{f(x_1,\ldots,x_n)\xrightarrow{\alpha} t}$, to transform r by replacing the aforementioned premise by H, y by t, and the x_i by the t_i; this is repeated (transfinitely) until all positive premises with a non-variable left-hand side have disappeared. The same idea will be applied in the proof of the forthcoming Proposition 3.4. In the final transformation step, rules with a *negative* conclusion $t \not\xrightarrow{\alpha}$ are introduced. The motivation is that instead of the notion of well-founded provability of Def. 2.11, we want a more constructive notion, like the one of Def. 2.10, in which proof step 2 is cast as a special case of proof step 1. A rule r with a conclusion $f(x_1, \ldots, x_n) \not\xrightarrow{\alpha}$ is obtained by picking one premise from each rule with a conclusion $f(x_1, \ldots, x_n) \xrightarrow{\alpha} t$, and including the denial of each of the selected premises as a premise of r. For this last transformation it is essential that rules do not have lookahead.

The resulting TSS, which is in pure ntyft format without lookahead, is denoted by P^+. In [5] it is established that $P^+ \vdash \alpha \Leftrightarrow P \vdash_{ws} \alpha$ for all closed literals α. The notion of irredundant provability is adapted in a straightforward fashion to accommodate rules with a negative conclusion. P-ruloids are the pure nxytt rules without lookahead that are irredundantly provable from P^+.

The following correspondence result from [5] between a TSS and its ruloids will play a crucial rôle in our decomposition method. It says that, provided we have enough variables to build ruloids, there is a well-supported proof from P of a transition $p \xrightarrow{a} q$, with p a closed substitution instance of a term t, if and only if there is a proof of this transition that uses at the root a P-ruloid with source t.

Theorem 3.2 (Lemma 8.2 from [5], which is Lemma 13 in the report version of [5]) *Let the set V of variables be infinite and satisfying $|V| \geq |A|$. Let $P = (\Sigma, R)$ be a TSS in ready simulation format with $|\Sigma| \leq |V|$. Then $P \vdash_{ws} \sigma(t) \xrightarrow{a} p$ for $t \in \mathbb{T}(\Sigma)$, $p \in T(\Sigma)$ and σ a closed substitution, if and only if there are a P-ruloid $\frac{H}{t \xrightarrow{a} u}$ and a closed substitution σ' with $P \vdash_{ws} \sigma'(\alpha)$ for $\alpha \in H$, $\sigma'(t) = \sigma(t)$ and $\sigma'(u) = p$.*

3.3 Ruloids for the Tyft/Tyxt Format

In this section we construct ruloids for TSSs in tyft/tyxt format (see Theorem 3.9). We proceed in three steps. First any TSS in tyft/tyxt format is converted into a TSS in pure xyft format, applying a result from [9] (Proposition 3.3). Then we show that the pure xytt rules irredundantly provable from the resulting TSS satisfy all properties of ruloids, except for having bounded lookahead (Proposition 3.4). Finally we get rid of unbounded lookahead (Proposition 3.7).

Proposition 3.3 ([9], Section 4) *Let the set V of variables be infinite. For each TSS P in tyft/tyxt format there is a TSS Q in pure xyft format, such that $P \vdash \alpha \Leftrightarrow Q \vdash \alpha$ for all closed positive literals α.*

Proposition 3.4 *Let the set V of variables be infinite and satisfying $|V| \geq |A|$. Let $P = (\Sigma, R)$ be a TSS in pure tyft format with $|\Sigma| \leq |V|$. Then $P \vdash \sigma(t) \xrightarrow{a} p$ for $t \in \mathbb{T}(\Sigma)$, $p \in T(\Sigma)$ and σ a closed substitution, if and only if there are a pure xytt rule $\frac{H}{t \xrightarrow{a} u}$ and a closed substitution σ' with $P \vdash_{\mathrm{irr}} \frac{H}{t \xrightarrow{a} u}$, $P \vdash \sigma'(\alpha)$ for $\alpha \in H$, $\sigma'(t) = \sigma(t)$ and $\sigma'(u) = p$.*

Before giving the proof of this proposition, we simultaneously illustrate Theorem 3.2 and Proposition 3.4 through a TSS in pure tyft format without lookahead. It is based on an example in [9].

Example 3.5 (A fragment of CCS with replication). *Let $\mathcal{A}$ be a set of* names. *The set $\bar{\mathcal{A}}$ of* co-names *is given by $\bar{\mathcal{A}} = \{\bar{a} \mid a \in \mathcal{A}\}$, and $\mathcal{L} = \mathcal{A} \cup \bar{\mathcal{A}}$ is the set of* visible actions. *The function $\bar{\cdot}$ is extended to $\mathcal{L}$ by declaring $\bar{\bar{a}} = a$. Furthermore $A = \mathcal{L} \cup \{\tau\}$ is the set of actions. Note that $\bar{\tau}$ is undefined.*

The process algebra CCS has a constant 0*, a unary operator a for $a \in A$, binary operators $+$ and $|$, and a few constructs that are omitted here. In addition we consider the unary* replication operator !. *The TSS* CCS! *is given by the pure*

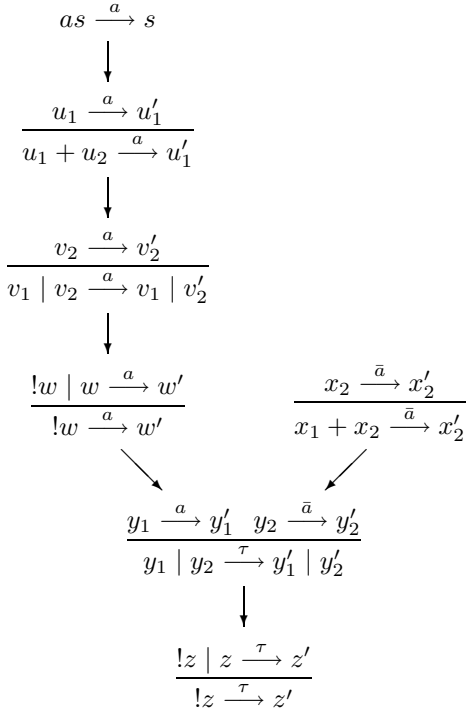
tyft rules without lookahead below; note that the rule for replication is not xyft. These rules are actually schemata, where a ranges over A.

$$ax \xrightarrow{a} x \qquad \frac{x \xrightarrow{a} x'}{x+y \xrightarrow{a} x'} \qquad \frac{y \xrightarrow{a} y'}{x+y \xrightarrow{a} y'}$$

$$\frac{x \xrightarrow{a} x'}{x \mid y \xrightarrow{a} x' \mid y} \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{\bar{a}} y'}{x \mid y \xrightarrow{\tau} x' \mid y'} \qquad \frac{y \xrightarrow{a} y'}{x \mid y \xrightarrow{a} x \mid y'} \qquad \frac{!x \mid x \xrightarrow{a} x'}{!x \xrightarrow{a} x'}$$

We give an example application of Proposition 3.4, with $t = !(a0+x)$, $\sigma(x) = \bar{a}0$ *and* $p = (a0 + \bar{a}0) \mid 0 \mid 0$. *Note that* $\sigma(t) \xrightarrow{\tau} p$ *is part of the transition relation.*

Below we depict the construction of an irredundant proof of the pure xytt rule $\dfrac{x \xrightarrow{\bar{a}} x'}{!(a0+x) \xrightarrow{\tau} !(a0+x) \mid 0 \mid x'}$. *Labels of the proof tree are obtained by applying* ρ *to the conclusions of the rules in the structure below. Definition 2.10 of a proof tree does not specify the identity of the nodes that occur in the tree; it merely poses requirements on their labels. Following a convention from [9], we do not depict the hypothesis explicitly, and any other node we take to be an* α*-conversion of the proof rule applied in that node to obtain its label from the labels of the nodes above it. By choosing the variables in all these rules to be disjoint, we can unite the substitutions applied in each of these nodes into one global substitution* ρ.



$$\begin{array}{lcl}
\rho(s) & = & 0 \\
\rho(u_1) & = & a0 \\
\rho(u_1') & = & 0 \\
\rho(u_2) & = & x \\
\rho(v_1) & = & !(a0+x) \\
\rho(v_2) & = & a0+x \\
\rho(v_2') & = & 0 \\
\rho(w) & = & a0+x \\
\rho(w') & = & !(a0+x) \mid 0 \\
\rho(x_1) & = & a0 \\
\rho(x_2) & = & x \\
\rho(x_2') & = & x' \\
\rho(y_1) & = & !(a0+x) \\
\rho(y_1') & = & !(a0+x) \mid 0 \\
\rho(y_2) & = & a0+x \\
\rho(y_2') & = & x' \\
\rho(z) & = & a0+x \\
\rho(z') & = & !(a0+x) \mid 0 \mid x'
\end{array}$$

Moreover, $\sigma'(x) = \bar{a}0$ *and* $\sigma'(x') = 0$. *Note that* $\sigma'(x \xrightarrow{\bar{a}} x') = \bar{a}0 \xrightarrow{\bar{a}} 0$ *is part of the transition relation.*

Proof of Proposition 3.4: $\boxed{\Leftarrow}$ Suppose there are a pure xytt rule $\frac{H}{t \xrightarrow{a} u}$ and a closed substitution σ', with $P \vdash_{\mathrm{irr}} \frac{H}{t \xrightarrow{a} u}$, $P \vdash \sigma'(\alpha)$ for $\alpha \in H$, $\sigma'(t) = \sigma(t)$ and $\sigma'(u) = p$. Let π be an irredundant proof from P of $\frac{H}{t \xrightarrow{a} u}$. Then a proof of $\sigma(t) \xrightarrow{a} p$ is obtained by taking $\sigma'(\pi)$ and replacing each $\sigma'(\alpha)$ for $\alpha \in H$ by a proof of $\sigma'(\alpha)$.

$\boxed{\Rightarrow}$ Let $P \vdash \sigma(t) \xrightarrow{a} p$. First, suppose t is a variable. Let y be a different variable. By definition, the pure xytt rule $\frac{t \xrightarrow{a} y}{t \xrightarrow{a} y}$ is irredundantly provable from P. Let σ' be a closed substitution with $\sigma'(t) = \sigma(t)$ and $\sigma'(y) = p$. Clearly, $P \vdash \sigma'(t \xrightarrow{a} y)$.

Next, suppose $t = f(t_1, \ldots, t_{ar(f)})$. We apply structural induction on a closed proof π from P of $\sigma(t) \xrightarrow{a} p$. Let $r \in R$ be the pure tyft rule and ρ with $dom(\rho) = var(r)$ be the closed substitution used at the bottom of π, where r is of the form $\frac{\{v_k \xrightarrow{c_k} y_k \mid k \in K\}}{f(x_1, \ldots, x_{ar(f)}) \xrightarrow{a} v}$. Then $\rho(x_i) = \sigma(t_i)$ for $i \in \{1, \ldots, ar(f)\}$, $\rho(v) = p$, and $\rho(v_k) \xrightarrow{c_k} \rho(y_k)$ for $k \in K$ are provable from P by means of strict subproofs of π.

We now show how to define two substitutions ρ_∞ and σ_∞, and a pure xytt rule $\frac{H}{t \xrightarrow{a} \rho_\infty(v)}$ that is irredundantly provable from P, such that $P \vdash \sigma_\infty(\alpha)$ for $\alpha \in H$ and $\sigma_\infty(t \xrightarrow{a} \rho_\infty(v)) = \sigma(t) \xrightarrow{a} p$. The substitutions ρ_∞ and σ_∞ will be given as the "limits" of sequences of substitutions ρ_d and σ_d, and the set H will be defined in the process. We begin by inductively defining substitutions ρ_d and σ_d for $d \in \mathbb{N}$, such that

$$dom(\rho_d) = \{z \in var(r) \mid depth(z) \leq d\}, \tag{1}$$

$$var(\rho_d(z)) \subseteq dom(\sigma_d) \text{ for } z \in dom(\rho_d), \text{ and} \tag{2}$$

$$\sigma_d \rho_d(z) = \rho(z) \text{ for } z \in dom(\rho_d). \tag{3}$$

Let ρ_0 and σ_0 be substitutions such that

$$dom(\rho_0) = \{x_1, \ldots, x_{ar(f)}\}, \tag{4}$$

$$dom(\sigma_0) = var(t), \tag{5}$$

$$\rho_0(x_i) = t_i \text{ for } i \in \{1, \ldots, ar(f)\}, \text{ and} \tag{6}$$

$$\sigma_0(z) = \sigma(z) \text{ for } z \in var(t). \tag{7}$$

Note that $\sigma_0(z) \in T(\Sigma)$ for $z \in dom(\sigma_0)$.

Properties (1)–(3) hold for $d = 0$: $dom(\rho_0) \overset{(4)}{=} \{z \in var(r) \mid depth(z) \leq 0\}$, $var(\rho_0(x_i)) \overset{(6)}{=} var(t_i) \subseteq var(t) \overset{(5)}{=} dom(\sigma_0)$ for $i \in \{1, \ldots, ar(f)\}$, and $\sigma_0 \rho_0(x_i) \overset{(6)}{=} \sigma_0(t_i) \overset{(7)}{=} \sigma(t_i) = \rho(x_i)$ for $i \in \{1, \ldots, ar(f)\}$.

Suppose ρ_d and σ_d have been defined for some $d \geq 0$. According to Definition 2.17, for $k \in K_d$ we have $depth(v_k) = d$, so $P \vdash \sigma_d\rho_d(v_k) \overset{(1),(3)}{=} \rho(v_k) \xrightarrow{c_k} \rho(y_k)$ by means of a strict subproof of π. By induction on the structure of π there are a pure xytt rule $\frac{H_k}{\rho_d(v_k) \xrightarrow{c_k} u_k}$ and a closed substitution σ'_k with

$$P \vdash_{\text{irr}} \frac{H_k}{\rho_d(v_k) \xrightarrow{c_k} u_k}, \tag{8}$$

$$P \vdash \sigma'_k(\alpha) \text{ for } \alpha \in H_k, \tag{9}$$

$$\sigma'_k\rho_d(v_k) = \sigma_d\rho_d(v_k), \text{ and} \tag{10}$$

$$\sigma'_k(u_k) = \rho(y_k). \tag{11}$$

We define the substitution ρ_{d+1} by:

$$dom(\rho_{d+1}) = dom(\rho_d) \cup \{y_k \in var(r) \mid k \in K_d\}, \tag{12}$$

$$\rho_{d+1}(z) = \rho_d(z) \text{ if } z \in dom(\rho_d), \text{ and} \tag{13}$$

$$\rho_{d+1}(y_k) = u_k \text{ for } k \in K_d. \tag{14}$$

Owing to the assumption that $|\Sigma| \leq |V|$ and $|A| \leq |V|$, we can choose the sets $rhs(H_k)$ for $k \in K_d$ pairwise disjoint, and disjoint from $dom(\sigma_d)$ (cf. Lemma 6.4 in [5], which is Lemma 6 in the report version of [5]). We define the substitution σ_{d+1} by:

$$dom(\sigma_{d+1}) = dom(\sigma_d) \cup \bigcup_{k \in K_d} rhs(H_k), \tag{15}$$

$$\sigma_{d+1}(z) = \sigma_d(z) \text{ if } z \in dom(\sigma_d), \text{ and} \tag{16}$$

$$\sigma_{d+1}(z) = \sigma'_k(z) \text{ if } z \in rhs(H_k) \text{ for some } k \in K_d. \tag{17}$$

Note that $\sigma_{d+1}(z) \in T(\Sigma)$ for $z \in dom(\sigma_{d+1})$.

We proceed to prove (1)–(3) for $d+1$, assuming by induction that they hold for d. First, $dom(\rho_{d+1}) \overset{(12),(1)}{=} \{z \in var(r) \mid depth(z) \leq d+1\}$. Next, we prove that $var(\rho_{d+1}(z)) \subseteq dom(\sigma_{d+1})$ for $z \in dom(\rho_{d+1})$.

- Let $z \in dom(\rho_d)$. Then $var(\rho_{d+1}(z)) \overset{(13)}{=} var(\rho_d(z)) \overset{(2)}{\subseteq} dom(\sigma_d) \overset{(15)}{\subseteq} dom(\sigma_{d+1})$.
- Let $z = y_k$ for some $k \in K_d$. Then $var(\rho_{d+1}(z)) \overset{(14)}{=} var(u_k) \subseteq var(\rho_d(v_k)) \cup rhs(H_k)$ by the pureness of $\frac{H_k}{\rho_d(v_k) \xrightarrow{c_k} u_k}$. We have $var(v_k) \overset{(1)}{\subseteq} dom(\rho_d)$, so $var(\rho_d(v_k)) \overset{(2)}{\subseteq} dom(\sigma_d) \overset{(15)}{\subseteq} dom(\sigma_{d+1})$. Further, $rhs(H_k) \overset{(15)}{\subseteq} dom(\sigma_{d+1})$.

Finally, we prove that $\sigma_{d+1}\rho_{d+1}(z) = \rho(z)$ for $z \in dom(\rho_{d+1})$.

- Let $z \in dom(\rho_d)$. Then $\sigma_{d+1}\rho_{d+1}(z) \overset{(13)}{=} \sigma_{d+1}\rho_d(z) \overset{(2),(16)}{=} \sigma_d\rho_d(z) \overset{(3)}{=} \rho(z)$.

- Let $z = y_k$ for some $k \in K_d$. Let $w \in var(\rho_d(v_k))$. Then $w \overset{(2)}{\in} dom(\sigma_d)$, and hence $\sigma_{d+1}(w) \overset{(16)}{=} \sigma_d(w)$. Since $\sigma'_k\rho_d(v_k) \overset{(10)}{=} \sigma_d\rho_d(v_k)$, it follows that $\sigma_d(w) = \sigma'_k(w)$. Hence

$$\sigma_{d+1}(w) = \sigma'_k(w), \text{ if } w \in var(\rho_d(v_k)) \text{ for some } k \in K_d. \tag{18}$$

  By the pureness of $\frac{H_k}{\rho_d(v_k) \xrightarrow{c_k} u_k}$, we have $var(u_k) \subseteq var(\rho_d(v_k)) \cup rhs(H_k)$, so $\sigma_{d+1}\rho_{d+1}(z) \overset{(14)}{=} \sigma_{d+1}(u_k) \overset{(17),(18)}{=} \sigma'_k(u_k) \overset{(11)}{=} \rho(z)$.

Hence, (1)-(3) hold for $d+1$.

Now we define a substitution ρ_∞ with $dom(\rho_\infty) = var(r)$ as the limit of the ρ_d, so

$$\rho_\infty(z) = \rho_d(z) \text{ for } z \in dom(\rho_d) \text{ and } d \in \mathbb{N}. \tag{19}$$

Let $H = \bigcup_{k \in K} H_k$. We define a closed substitution σ_∞ that behaves as the limit of the σ_d on variables from $var(t) \cup rhs(H)$ and maps all other variables to arbitrary closed terms.

$$\sigma_\infty(z) = \sigma_d(z) \text{ for } z \in dom(\sigma_d) \text{ and } d \in \mathbb{N}. \tag{20}$$

The substitutions ρ_∞ and σ_∞ are well-defined, owing to (13) and (16). We verify that the rule $\frac{H}{t \xrightarrow{a} \rho_\infty(v)}$ together with the closed substitution σ_∞ satisfies the desired properties.

- $\frac{H}{t \xrightarrow{a} \rho_\infty(v)}$ is pure xytt:

  The right-hand sides of the positive premises in any H_k are distinct variables. By construction, these sets of variables (one for every $k \in K$) are pairwise disjoint, and disjoint from $var(t)$. Furthermore, the left-hand sides of the premises in any H_k are variables. This makes $\frac{H}{t \xrightarrow{a} \rho_\infty(v)}$ an xytt rule.

  We prove that this rule does not contain free variables. We consider two cases:

  - z is the left-hand side of a premise in H_k for some $k \in K$ and $z \notin rhs(H)$. Since $\frac{H_k}{\rho_\infty(v_k) \xrightarrow{c_k} u_k}$ is pure, $z \in var(\rho_\infty(v_k))$.
  - $z \in var(\rho_\infty(v))$.

  Since $var(\rho_\infty(w)) \overset{(2)}{\subseteq} dom(\sigma_\infty) \overset{(5),(15)}{=} var(t) \cup rhs(H)$ for $w \in var(r)$, in either case $z \in var(t) \cup rhs(H)$.

We prove that $\frac{H}{t\xrightarrow{a}\rho_\infty(v)}$ is well-founded. Let $y \in lhs(H_k) \cap rhs(H_\ell)$ with $k \in K_d$, $\ell \in K_e$ and $k \neq \ell$. We prove that $e < d$. Since $\frac{H}{t\xrightarrow{a}\rho_\infty(v)}$ is xytt and $y \in rhs(H_\ell)$, it follows that $y \notin rhs(H_k)$ and $y \notin var(t)$. Since $\frac{H_k}{\rho_\infty(v_k)\xrightarrow{c_k}u_k}$ is pure and $y \in lhs(H_k)\backslash rhs(H_k)$, we have $y \in var(\rho_\infty(v_k))$. This implies that $y \in var(\rho_d(v_k)) \overset{(2)}{\subseteq} dom(\sigma_d) \overset{(5),(15)}{\subseteq} var(t) \cup \bigcup_{k' \in K_{d-1}\cup\ldots\cup K_0} rhs(H_{k'})$. Since $y \notin var(t)$ and $y \in rhs(H_\ell)$ with $\ell \in K_e$ we have $e \in \{0, \ldots, d-1\}$ and thus $e < d$. Thus, if $\langle x, y\rangle$ and $\langle y, z\rangle$ are edges in the dependency graphs of respectively H_ℓ with $\ell \in K_e$ and H_k with $k \in K_d$ and $k \neq \ell$, then $e < d$. Moreover, well-foundedness of the rules $\frac{H_k}{\rho_\infty(v_k)\xrightarrow{c_k}u_k}$ for $k \in K$ implies that their dependency graphs do not contain infinite backward chains. Hence, the dependency graph of H does not contain an infinite backwards chain of edges either. Therefore, $\frac{H}{t\xrightarrow{a}\rho_\infty(v)}$ is well-founded.

- $P \vdash_{\mathrm{irr}} \frac{H}{t\xrightarrow{a}\rho_\infty(v)}$:

  Since $r \in R$, we have $P \vdash_{\mathrm{irr}} \rho_\infty(r) \overset{(6),(14)}{=} \frac{\{\rho_\infty(v_k)\xrightarrow{c_k}u_k \mid k\in K\}}{t\xrightarrow{a}\rho_\infty(v)}$. Furthermore, $P \overset{(8)}{\vdash}_{\mathrm{irr}} \frac{H_k}{\rho_\infty(v_k)\xrightarrow{c_k}u_k}$ for $k \in K$. As $H = \bigcup_{k\in K} H_k$, it follows that $P \vdash_{\mathrm{irr}} \frac{H}{t\xrightarrow{a}\rho_\infty(v)}$.

- $P \vdash \sigma_\infty(\alpha)$ for $\alpha \in H$:

  Let $\alpha \in H$. Then there is a $k \in K$ such that $\alpha \in H_k$, so $k \in K_d$ for some $d \geq 0$. By the pureness of $\frac{H_k}{\rho_d(v_k)\xrightarrow{c_k}u_k}$, it follows that $var(\alpha) \subseteq var(\rho_d(v_k)) \cup var(rhs(H_k))$, so $P \overset{(9)}{\vdash} \sigma'_k(\alpha) \overset{(17),(18)}{=} \sigma_{d+1}(\alpha)) \overset{(20)}{=} \sigma_\infty(\alpha)$.

- $\sigma_\infty(t) \overset{(7)}{=} \sigma(t)$.

- $\sigma_\infty\rho_\infty(v) \overset{(3)}{=} \rho(v) = p$. □

We have now obtained the required ruloids, except that we still have to get rid of unbounded lookahead. We will do this by replacing any ruloid by one with bounded lookahead.

definition 3.6 (replacement) *We say that a transition rule r* can be replaced by *a transition rule s if r is a substitution instance of s and for any closed substitution instance $\frac{K}{\alpha}$ of s there is a closed substitution instance $\frac{H}{\alpha}$ of r such that $H \subseteq K$.*

Note that if a transition rule r can be replaced by a transition rule s, the TSSs $(\Sigma, R \cup \{r\})$ and $(\Sigma, R \cup \{s\})$ specify the same transition relation.

The following proposition requires an abundance of variables. Whereas for Theorem 3.2 and Propositions 3.3 and 3.4 we merely needed infinitely many

variables and $|V| \geq |A| + |\Sigma|$, here we need uncountably many variables and $|V| > |\Sigma|$, so that we have $|V| > |T(\Sigma)|$.

Proposition 3.7 *Suppose V is uncountable and $|\Sigma| < |V|$. Any pure xytt rule r can be replaced by a pure xytt rule s with bounded lookahead with the same conclusion.*

Proof: Let $r = \frac{H}{\alpha}$. Let κ be an infinite cardinal, such that $|T(\Sigma)| < \kappa \leq |V|$, and let $\mathcal{O}_\kappa$ denote the set of sequences of strictly decreasing ordinals that have a cardinality smaller than κ. Let $V(r)$ denote the finite set of variables y_0 such that r has premises $y_0 \xrightarrow{a_1} y_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} y_n$ for $n \geq 0$ with y_n occurring in α, and let $U(r)$ be the set of all other variables occurring in r. For every variable $x \in U(r)$ and $\eta \in \mathcal{O}_\kappa$, pick a distinct variable $x_\eta \in V$ not occurring in r. This is possible because $|\mathcal{O}_\kappa| = \kappa$. Let $s = \frac{K}{\alpha}$ with

$$\begin{aligned} K = \quad & \{x \xrightarrow{a} y \mid (x \xrightarrow{a} y) \in H,\ y \in V(r)\} \cup \\ & \{x \xrightarrow{a} y_\lambda \mid (x \xrightarrow{a} y) \in H,\ x \in V(r),\ y \in U(r),\ \lambda < \kappa\} \cup \\ & \{x_\eta \xrightarrow{a} y_{\eta\lambda} \mid (x \xrightarrow{a} y) \in H,\ x \in U(r),\ \eta, \eta\lambda \in \mathcal{O}_\kappa\}. \end{aligned}$$

By definition s is an xytt rule, and if r is pure then so is s. As there are only finitely many variables occurring in α, and there is no infinite chain of strictly decreasing ordinals, the rule s has bounded lookahead. It remains to be shown that r can be replaced by s.

First of all the substitution ξ given by $\xi(x_\eta) = x$ for all $x \in U(r)$ and $\eta \in \mathcal{O}_\kappa$ shows that r is a substitution instance of s.

Now consider a closed substitution σ. We need to define a closed substitution ρ, such that $\rho(H) \subseteq \sigma(K)$ and $\rho(\alpha) = \sigma(\alpha)$. For $x \in V(r)$ let $\rho(x) = \sigma(x)$. Since $x \in V(r)$ for $x \in var(\alpha)$, $\rho(\alpha) = \sigma(\alpha)$. Define the *relative depth* $rd(x)$ of variables $x \in var(r)$ by

$$\begin{aligned} rd(x) &= 0 && \text{for } x \in V(r) \\ rd(y) &= rd(x) + 1 && \text{for } y \in U(r) \text{ and } (x \xrightarrow{a} y) \in H. \end{aligned}$$

As in Definition 2.17 this is well-defined because r is well-founded. With induction on the relative depth of $x \in U(r)$ we are going to define $\rho(x)$ and $S_x \subseteq \{\eta \in \mathcal{O}_\kappa \mid \sigma(x_\eta) = \rho(x)\}$ such that $|\{\lambda < \kappa \mid \exists \eta \in \mathcal{O}_\kappa : \eta\lambda \in S_x\}| = \kappa$.
Base case: Let $x \in V(r)$, $y \in U(r)$ and $(x \xrightarrow{a} y) \in H$. As $|T(\Sigma)| < \kappa$, there must be a set $S \subseteq \{\lambda \mid \lambda < \kappa\}$ with $|S| = \kappa$ and $\sigma(y_\lambda) = \sigma(y_{\lambda'})$ for $\lambda, \lambda' \in S$. Let $\rho(y) = \sigma(y_\lambda)$ for some $\lambda \in S$. We define $S_y = S$. Then we have $\rho(x \xrightarrow{a} y) = \sigma(x \xrightarrow{a} y_\lambda) \in \sigma(K)$ for $\lambda \in S_y$ and $|\{\lambda < \kappa \mid \exists \eta \in \mathcal{O}_\kappa : \eta\lambda \in S_y\}| = |S_y| = \kappa$.
Induction step: Let $x \in U(r)$, $\rho(x)$ is already defined, and $(x \xrightarrow{a} y) \in H$. As $|\{\lambda < \kappa \mid \exists \eta \in \mathcal{O}_\kappa : \eta\lambda \in S_x\}| = \kappa$ and hence $|\{\nu \mid \exists \eta\lambda \in S_x : \nu < \lambda\}| = \kappa$, whereas $|T(\Sigma)| < \kappa$, there must be a set $S' \subseteq \{\eta\nu \in \mathcal{O}_\kappa \mid \eta \in S_x\}$ with $|\{\nu < \kappa \mid \exists \eta \in \mathcal{O}_\kappa : \eta\nu \in S'\}| = \kappa$ and $\sigma(y_\eta) = \sigma(y_{\eta'})$ for $\eta, \eta' \in S'$. Let $\rho(y) = \sigma(y_{\eta\nu})$ for some $\eta\nu \in S'$. We define $S_y = S'$. Then we have $\rho(x \xrightarrow{a} y) = \sigma(x_\eta \xrightarrow{a} y_{\eta\nu}) \in \sigma(K)$ for $\eta\nu \in S_y$. Hence, $\rho(H) \subseteq \sigma(K)$. So r can be replaced by s. □

In the following definition of a ruloid and the forthcoming Theorems 3.9 and 3.11 we assume the existence of sufficiently many variables (satisfying the conditions of Propositions 3.4 and 3.7). However, this assumption will be discharged in Section 3.5, showing that our mail result (Theorem 3.11) will hold regardless.

definition 3.8 (ruloid) *Let P be a TSS in tyft/tyxt format. According to Proposition 3.3 there is a TSS Q in pure xyft format, such that $P \vdash \alpha \Leftrightarrow Q \vdash \alpha$ for all closed positive literals α. The set of* P-ruloids *is obtained by first taking the set of pure xytt rules, irredundantly provable from Q, and next replacing each of these pure xytt rules by one with bounded lookahead, using Proposition 3.7.*

Theorem 3.9 *Suppose V is uncountable and $|A| \leq |V|$. Let $P = (\Sigma, R)$ be a TSS in tyft/tyxt format with $|\Sigma| < |V|$. Then $P \vdash \sigma(t) \xrightarrow{a} p$ for $t \in \mathbb{T}(\Sigma)$, $p \in T(\Sigma)$ and σ a closed substitution, if and only if there are a P-ruloid $\frac{H}{t \xrightarrow{a} u}$ and a closed substitution σ' with $P \vdash \sigma'(\alpha)$ for $\alpha \in H$, $\sigma'(t) = \sigma(t)$ and $\sigma'(u) = p$.*

Proof: According to Proposition 3.3 there is a TSS Q in pure xyft format, such that $P \vdash \alpha \Leftrightarrow Q \vdash \alpha$ for all closed positive literals α.

$\boxed{\Rightarrow}$ Suppose $P \vdash \sigma(t) \xrightarrow{a} p$. Then $Q \vdash \sigma(t) \xrightarrow{a} p$. According to Proposition 3.4 ($\Rightarrow$), there are a pure xytt rule $\frac{G}{t \xrightarrow{a} u}$ and a closed substitution σ' with $Q \vdash_{\rm irr} \frac{G}{t \xrightarrow{a} u}$, $Q \vdash \sigma'(\alpha)$ for $\alpha \in G$, $\sigma'(t) = \sigma(t)$ and $\sigma'(u) = p$. Replace $\frac{G}{t \xrightarrow{a} u}$ by the P-ruloid $\frac{H}{t \xrightarrow{a} u}$, using Proposition 3.7. According to Definition 3.6, there is a substitution σ'' such that $\sigma''(\frac{H}{t \xrightarrow{a} u}) = \frac{G}{t \xrightarrow{a} u}$. Hence $Q \vdash \sigma'\sigma''(\alpha)$ for $\alpha \in H$, $\sigma''(t) = t$ and $\sigma''(u) = u$. Thus for the P-ruloid $\frac{H}{t \xrightarrow{a} u}$ and the closed substitution $\sigma'\sigma''$ we have $P \vdash \sigma'\sigma''(\alpha)$ for $\alpha \in H$, $\sigma'\sigma''(t) = \sigma'(t) = \sigma(t)$ and $\sigma'\sigma''(u) = \sigma'(u) = p$.

$\boxed{\Leftarrow}$ Suppose there are a P-ruloid $\frac{H}{t \xrightarrow{a} u}$ and a closed substitution σ' with $P \vdash \sigma'(\alpha)$ for $\alpha \in H$, $\sigma'(t) = \sigma(t)$ and $\sigma'(u) = p$. Then $Q \vdash \sigma'(\alpha)$ for $\alpha \in H$. $\frac{H}{t \xrightarrow{a} u}$ is a replacement of a pure xytt rule $\frac{G}{t \xrightarrow{a} u}$ irredundantly provable from Q. According to Definition 3.6, for $\sigma'(\frac{H}{t \xrightarrow{a} u})$ there is a closed substitution σ'' such that $\sigma''(G) \subseteq \sigma'(H)$, $\sigma''(t) = \sigma'(t)$ and $\sigma''(u) = \sigma'(u)$. Since $Q \vdash \sigma'(\alpha)$ for $\alpha \in H$, we have $Q \vdash \sigma''(\alpha)$ for $\alpha \in G$. Thus according to Proposition 3.4 ($\Leftarrow$) we have $Q \vdash \sigma(t) \xrightarrow{a} p$, so $P \vdash \sigma(t) \xrightarrow{a} p$. □

3.4 Decomposition of HML Formulas

Given a TSS $P = (\Sigma, R)$ in tyft/tyxt format or in ready simulation format, the following definition assigns to each term $t \in \mathbb{T}(\Sigma)$ and each observation $\varphi \in \mathbb{O}$ a collection $t_P^{-1}(\varphi)$ of decomposition mappings $\psi : V \to \mathbb{O}$. Each of these mappings $\psi \in t_P^{-1}(\varphi)$ guarantees, given a closed substitution σ, that $\sigma(t)$ satisfies φ if $\sigma(x)$ satisfies the formula $\psi(x)$ for all $x \in var(t)$. Moreover, whenever for some closed substitution σ the term $\sigma(t)$ satisfies φ, there must be a $\psi \in t_P^{-1}(\varphi)$ with $\sigma(x)$ satisfying $\psi(x)$ for all $x \in var(t)$. This is formalised in Theorem 3.11.

definition 3.10 *Let $P = (\Sigma, R)$ be a TSS in tyft/tyxt format (resp. ready simulation format) and assume that the set V of variables is infinite, with $|V| \geq |A|$ and $|V| > |\Sigma|$. Then $\cdot_P^{-1} : \mathbb{T}(\Sigma) \to (\mathbb{O} \to \mathcal{P}(V \to \mathbb{O}))$ is defined by:*

- *$\psi \in t_P^{-1}(\langle a \rangle \varphi)$ iff there is a P-ruloid $\frac{H}{t \xrightarrow{a} u}$ and a $\chi \in u_P^{-1}(\varphi)$ and $\psi : V \to \mathbb{O}$ is given by*

$$\psi(x) = \begin{cases} \bigwedge_{(x \xrightarrow{b} y) \in H} \langle b \rangle \psi(y) \ \wedge \bigwedge_{(x \xrightarrow{c} /) \in H} \neg \langle c \rangle \top \ \wedge \ \chi(x) & \text{if } x \in var(u) \\ \bigwedge_{(x \xrightarrow{b} y) \in H} \langle b \rangle \psi(y) \ \wedge \bigwedge_{(x \xrightarrow{c} /) \in H} \neg \langle c \rangle \top & \text{if } x \notin var(u) \end{cases}$$

- *$\psi \in t_P^{-1}(\bigwedge_{i \in I} \varphi_i)$ iff*

$$\psi(x) = \bigwedge_{i \in I} \psi_i(x)$$

  where $\psi_i \in t_P^{-1}(\varphi_i)$ for $i \in I$.

- *$\psi \in t_P^{-1}(\neg \varphi)$ iff there is a function $h : t_P^{-1}(\varphi) \to var(t)$ and $\psi : V \to \mathbb{O}$ is given by*

$$\psi(x) = \bigwedge_{\chi \in h^{-1}(x)} \neg \chi(x)$$

To explain the idea behind Definition 3.10, we expand on two of its cases. Consider $t_P^{-1}(\langle a \rangle \varphi)$, and let σ be any closed substitution. The question is under which conditions $\psi(x) \in \mathbb{O}$ on $\sigma(x)$, for $x \in var(t)$, there is a transition $\sigma(t) \xrightarrow{a} q$ with $q \models \varphi$. According to Theorems 3.2 and 3.9, there is such a transition if and only if there is a closed substitution σ' with $\sigma'(t) = \sigma(t)$ and a P-ruloid $\frac{H}{t \xrightarrow{a} u}$ such that (1) the premises in $\sigma'(H)$ are satisfied and (2) $\sigma'(u) \models \varphi$. The first condition is covered if for $x \in var(t)$, $\psi(x)$ contains conjuncts $\langle b \rangle \psi(y)$ for $x \xrightarrow{b} y \in H$ and conjuncts $\neg \langle c \rangle \top$ for $x \xrightarrow{c} / \in H$. By adding, for some $\chi \in u^{-1}(\varphi)$, a conjunct $\chi(x)$ if $x \in var(u)$, the second condition is covered as well.

Consider $t_P^{-1}(\neg \varphi)$, and let σ be any closed substitution. We have $\sigma(t) \not\models \varphi$ if and only if there is no $\chi \in t^{-1}(\varphi)$ such that $\sigma(x) \models \chi(x)$ for all $x \in var(t)$. In other words, for each $\chi \in t^{-1}(\varphi)$, $\psi(x)$ must contain a conjunct $\neg \chi(x)$, for some $x \in var(t)$.

Each $\psi(x)$ in Definition 3.10 is a HML formula, owing to the fact that the rule $\frac{H}{t \xrightarrow{a} u}$ in the definition of $t_P^{-1}(\langle a \rangle \varphi)$ has bounded lookahead. When clear from the context, subscripts P will be omitted.

Theorem 3.11 *Let $P = (\Sigma, R)$ be a complete TSS in tyft/tyxt format (resp. ready simulation format). For any $t \in \mathbb{T}(\Sigma)$, $\sigma : V \to T(\Sigma)$ and $\varphi \in \mathbb{O}$:*

$$\sigma(t) \models_P \varphi \quad \Leftrightarrow \quad \exists \psi \in t_P^{-1}(\varphi) \, \forall x \in var(t) : \sigma(x) \models_P \psi(x)$$

Proof: With induction on the structure of φ. Here we assume that V is infinite, $|V| \geq |A|$ and $|V| > |\Sigma|$. This assumption will be discharged in Section 3.5.

- $\varphi = \langle a \rangle \varphi'$

  $\boxed{\Rightarrow}$ Suppose $\sigma(t) \models \langle a \rangle \varphi'$. Then by Definition 2.3 there is a $p \in T(\Sigma)$ with $P \vdash_{ws} \sigma(t) \xrightarrow{a} p$ and $p \models \varphi'$. Thus, by Theorem 3.9 (resp. Theorem 3.2) there must be a P-ruloid $\frac{H}{t \xrightarrow{a} u}$ and a closed substitution σ' with $P \vdash_{ws} \sigma'(\alpha)$ for $\alpha \in H$, $\sigma'(t) = \sigma(t)$, i.e. $\sigma'(x) = \sigma(x)$ for $x \in var(t)$, and $\sigma'(u) = p$. Since $\sigma'(u) \models \varphi'$, the induction hypothesis can be applied, and there must be a $\chi \in u^{-1}(\varphi')$ such that $\sigma'(z) \models \chi(z)$ for all $z \in var(u)$. Now define $\psi \in t^{-1}(\langle a \rangle \varphi')$ as indicated in Definition 3.10, using $\frac{H}{t \xrightarrow{a} u}$ and χ. For $(x \xrightarrow{b} y) \in H$ one has $P \vdash_{ws} \sigma'(x) \xrightarrow{b} \sigma'(y)$. Moreover, for $(x \xrightarrow{c} \not\rightarrow) \in H$ one has $P \vdash_{ws} \sigma'(x) \xrightarrow{c} \not\rightarrow$, so the consistency of $\vdash_{ws}$ yields $P \not\vdash_{ws} \sigma'(x) \xrightarrow{c} q$ for all $q \in T(\Sigma)$, and thus $\sigma'(x) \models \neg\langle c \rangle \top$. We proceed to prove that $\sigma'(x) \models \psi(x)$ for $x \in V$. We apply ordinal induction on the *lookahead of x in H*, which is defined by

  $$\sup\{1 + \text{lookahead of } y \text{ in } H \mid x \xrightarrow{b} y \in H \text{ for some } b \in A\}$$

  Since $\frac{H}{t \xrightarrow{a} u}$ has bounded lookahead, the lookahead of x in H is a well-defined ordinal number. We distinguish two cases.

  - $x \in var(u)$. Then $\psi(x) = \bigwedge_{(x \xrightarrow{b} y) \in H} \langle b \rangle \psi(y) \wedge \bigwedge_{(x \xrightarrow{c} \not\rightarrow) \in H} \neg\langle c \rangle \top \wedge \chi(x)$. By induction on lookahead $\sigma'(y) \models \psi(y)$, so $\sigma'(x) \models \langle b \rangle \psi(y)$ for $(x \xrightarrow{b} y) \in H$. Since moreover $\sigma'(x) \models \neg\langle c \rangle \top$ for $(x \xrightarrow{c} \not\rightarrow) \in H$ and $\sigma'(x) \models \chi(x)$, it follows that $\sigma'(x) \models \psi(x)$.
  - $x \notin var(u)$. Then $\psi(x) = \bigwedge_{(x \xrightarrow{b} y) \in H} \langle b \rangle \psi(y) \wedge \bigwedge_{(x \xrightarrow{c} \not\rightarrow) \in H} \neg\langle c \rangle \top$. By induction on lookahead $\sigma'(y) \models \psi(y)$, so $\sigma'(x) \models \langle b \rangle \psi(y)$ for $(x \xrightarrow{b} y) \in H$. Since moreover $\sigma'(x) \models \neg\langle c \rangle \top$ for $(x \xrightarrow{c} \not\rightarrow) \in H$, it follows that $\sigma'(x) \models \psi(x)$.

  Finally, $\sigma(x) = \sigma'(x) \models \psi(x)$ for $x \in var(t)$.

  $\boxed{\Leftarrow}$ Suppose that there is a $\psi \in t^{-1}(\langle a \rangle \varphi')$ such that $\sigma(x) \models \psi(x)$ for all $x \in var(t)$. Let ψ be defined in terms of the P-ruloid $\frac{H}{t \xrightarrow{a} u}$ and $\chi \in u^{-1}(\varphi')$. We define a closed substitution σ' with as domain the variables in this P-ruloid such that:

  $$\sigma'(t) = \sigma(t), \tag{21}$$

  $$P \vdash_{ws} \sigma'(x) \xrightarrow{b} \sigma'(y) \text{ for } (x \xrightarrow{b} y) \in H, \tag{22}$$

  $$P \vdash_{ws} \sigma'(x) \xrightarrow{c} \not\rightarrow \text{ for } (x \xrightarrow{c} \not\rightarrow) \in H; \text{ and} \tag{23}$$

$$\sigma'(y) \models \psi(y) \text{ for } y \in dom(\sigma'). \tag{24}$$

$\sigma'(y)$ is defined by induction on the depth of y in the P-ruloid.

- $depth(y) = 0$. Then $y \in var(t)$. We define $\sigma'(y) = \sigma(y)$. This yields property (21). Moreover, property (24) for y holds. The formula $\psi(y)$ contains the conjunct $\neg\langle c\rangle\top$ for $(y \stackrel{c}{\not\longrightarrow}) \in H$, so by the completeness of P, property (24) for y implies property (23) for y.
- $depth(y) > 0$. Let $x \stackrel{b}{\longrightarrow} y$ be the positive premise in the P-ruloid with right-hand side y. By induction, $\sigma'(x)$ is already defined. By property (24), $\sigma'(x) \models \psi(x)$, so $\sigma'(x) \models \langle b\rangle\psi(y)$. Hence, $P \vdash_{ws} \sigma'(x) \stackrel{b}{\longrightarrow} p$ for some $p \in T(\Sigma)$ with $p \models \psi(y)$. We define $\sigma'(y) = p$. Then property (22) for the premise $x \stackrel{b}{\longrightarrow} y$ and property (24) for y hold. By the completeness of P, property (24) for y implies that property (23) for y holds as well.

Since $\psi(z)$ implies $\chi(z)$ for $z \in var(u)$, property (24) yields $\sigma'(y) \models \chi(y)$ for $y \in var(u)$. Thus the induction hypothesis can be applied, so $\sigma'(u) \models \varphi'$. By properties (21)–(23) together with Theorem 3.9 (resp. Theorem 3.2), $P \vdash_{ws} \sigma(t) \stackrel{a}{\longrightarrow} \sigma'(u)$. Hence, $\sigma(t) \models \langle a\rangle\varphi'$.

- $\varphi = \bigwedge_{i\in I} \varphi_i$
  $\sigma(t) \models \bigwedge_{i\in I} \varphi_i \;\Leftrightarrow\; \forall i{\in}I : \sigma(t) \models \varphi_i$
  $\Leftrightarrow\; \forall i{\in}I\; \exists \psi_i{\in}t^{-1}(\varphi_i)\; \forall x{\in}var(t) : \sigma(x) \models \psi_i(x)$
  $\Leftrightarrow\; \exists \psi{\in}t^{-1}(\bigwedge_{i\in I} \varphi_i)\; \forall x{\in}var(t) : \sigma(x) \models \psi(x).$
- $\varphi = \neg\varphi'$
  $\sigma(t) \models \neg\varphi' \;\Leftrightarrow\; \sigma(t) \not\models \varphi'$
  $\Leftrightarrow\; \exists h : t^{-1}(\varphi') \to var(t)\; \forall \chi \in t^{-1}(\varphi') : \sigma(h(\chi)) \not\models \chi(h(\chi))$
  $\Leftrightarrow\; \exists h : t^{-1}(\varphi') \to var(t)\; \forall x \in var(t) : \sigma(x) \models \bigwedge_{\chi\in h^{-1}(x)} \neg\chi(x)$
  $\Leftrightarrow\; \exists \psi \in t^{-1}(\neg\varphi')\; \forall x \in var(t) : \sigma(x) \models \psi(x).$ □

3.5 Counting variables

So far we have developed a modal decomposition method that applies to arbitrary languages with a structural operational semantics in tyft/tyxt format or in ready simulation format, provided that we have an uncountable set of variables whose cardinality exceeds the cardinality of the collection of operators in the language, and at least equals the cardinality of the set of actions. In computer science applications, however, it is customary to work with countable sets of variables. Here we show that our decomposition result applies regardless of the cardinality of the set of variables.

Assume a given set of *true variables* V_{true}. These are the only ones that are used in a given application. Now, given a set of actions A and a TSS $P = (\Sigma, R)$, we *define* an uncountable set $V \supseteq V_{true}$ satisfying $|V| \geq |A|$ and $|V| > |\Sigma|$. The

elements of V are called *variables* and variables in $V - V_{true}$ are *virtual variables*. As long as only the true variables appear in TSSs and specifications of processes, nothing stops us from defining any amount of virtual variables to be used in our constructions. Terms, TSSs, ruloids and other syntactic constructions are called *true* if they only contain true variables, and *virtual* otherwise.

Theorems 3.2 and 3.9 (our intermediate results) apply merely to virtual ruloids; they may fail to hold for true ruloids. Definition 3.10 on the other hand provides a modal decomposition $t^{-1}(\varphi)$ for any virtual term t (and any HML formula φ), and thus surely for every true term t. Such a modal decomposition consists of a set of decomposition mappings ψ, each of which allocates an HML formula $\psi(x)$ to every variable x. As only the $\psi(x)$ for $x \in var(t)$ play a rôle in Theorem 3.11, nothing is lost by restricting the domain of $\psi(x)$ to $var(t)$. (This is an *a posteriori* restriction; virtual variables may have been used in the definition of $\psi(x)$.) Using this restriction of the decomposition mappings, Theorem 3.11 restricts to a valid statement involving true variables only.

3.6 Examples

We give two examples of the application of Theorem 3.11.

Example 3.12 *Let $A = \{a, b\}$ and $P = (\Sigma, R)$, where Σ consists of the constant c and the binary function symbol f, and R is:*

$$c \xrightarrow{a} c \qquad \frac{x_1 \xrightarrow{a} y}{f(x_1, x_2) \xrightarrow{b} y} \qquad \frac{x_2 \xrightarrow{a} y \quad x_1 \not\xrightarrow{b}}{f(x_1, x_2) \xrightarrow{b} y}$$

This TSS is complete and in ready simulation format. We proceed to compute $f(x_1, x_2)^{-1}(\langle b\rangle\top)$. The only two P-ruloids (modulo α-conversion) with a conclusion $f(x_1, x_2) \xrightarrow{b} \_$ are

$$\frac{x_1 \xrightarrow{a} y}{f(x_1, x_2) \xrightarrow{b} y} \qquad \frac{x_2 \xrightarrow{a} y \quad x_1 \not\xrightarrow{b}}{f(x_1, x_2) \xrightarrow{b} y}$$

According to Definition 3.10, $f(x_1, x_2)^{-1}(\langle b\rangle\top) = \{\psi_1, \psi_2\}$ where ψ_1 and ψ_2 are defined as follows, using some $\chi \in y^{-1}(\top)$ (so $\chi(x) = \top$ for all $x \in V$):

$$\begin{array}{ll} \psi_1(y) = \chi(y) = \top & \psi_2(y) = \chi(y) = \top \\ \psi_1(x_1) = \langle a\rangle\psi_1(y) = \langle a\rangle\top & \psi_2(x_1) = \neg\langle b\rangle\top \\ \psi_1(z) = \top \text{ if } z \notin \{y, x_1\} & \psi_2(x_2) = \langle a\rangle\psi_2(y) = \langle a\rangle\top \\ & \psi_2(z) = \top \text{ if } z \notin \{y, x_1, x_2\} \end{array}$$

So by Theorem 3.11 a closed term $f(p_1, p_2)$ can execute a b if and only if p_1 can execute an a, or p_1 cannot execute a b and p_2 can execute an a.

We proceed to compute $f(x_1, x_2)^{-1}(\neg\langle b\rangle\top)$. There are four possible functions $h : f(x_1, x_2)^{-1}(\langle b\rangle\top) \to var(f(x_1, x_2))$, yielding four possible definitions of $\psi \in f(x_1, x_2)^{-1}(\neg\langle b\rangle\top)$. In all four cases, $\psi(z) = \top$ for $z \notin \{x_1, x_2\}$.

1. *If* $h(\psi_1) = h(\psi_2) = x_1$ *then*

$$\psi(x_1) = \neg\psi_1(x_1) \wedge \neg\psi_2(x_1) = \neg\langle a\rangle\top \wedge \neg\neg\langle b\rangle\top = \neg\langle a\rangle\top \wedge \langle b\rangle\top$$
$$\psi(x_2) = \top$$

2. *If* $h(\psi_1) = h(\psi_2) = x_2$ *then*

$$\psi(x_1) = \top$$
$$\psi(x_2) = \neg\psi_1(x_2) \wedge \neg\psi_2(x_2) = \neg\top \wedge \neg\langle a\rangle\top$$

3. *If* $h(\psi_1) = x_1$ *and* $h(\psi_2) = x_2$ *then*

$$\psi(x_1) = \neg\psi_1(x_1) = \neg\langle a\rangle\top$$
$$\psi(x_2) = \neg\psi_2(x_2) = \neg\langle a\rangle\top$$

4. *If* $h(\psi_1) = x_2$ *and* $h(\psi_2) = x_1$ *then*

$$\psi(x_1) = \neg\psi_2(x_1) = \neg\neg\langle b\rangle\top = \langle b\rangle\top$$
$$\psi(x_2) = \neg\psi_1(x_2) = \neg\top$$

By Theorem 3.11 a closed term $f(p_1, p_2)$ *can*not *execute a b if and only if either (1)* p_1 *can execute a b but not an a, or (3) neither* p_1 *nor* p_2 *can execute an a. The other two possibilities (2) and (4) do not qualify, since no term can ever satisfy* $\neg\top$*.*

Example 3.13 *Let* $A = \{a, b\}$ *and* $P = (\Sigma, R)$*, where* Σ *consists of the constant* c *and the unary function symbol* f*, and* R *is:*

$$c \xrightarrow{a} c \qquad \frac{x \xrightarrow{a} y}{f(x) \xrightarrow{b} y} \qquad \frac{x \xrightarrow{b} y \quad y \xrightarrow{a} z}{f(x) \xrightarrow{a} f(y)}$$

This TSS is complete and in tyft/tyxt format. We compute $f(f(x))^{-1}(\langle b\rangle\langle a\rangle\top)$*. The only P-ruloid (modulo* α*-conversion) with a conclusion* $f(f(x)) \xrightarrow{b} \_$ *is*

$$\frac{x \xrightarrow{b} y \quad y \xrightarrow{a} z}{f(f(x)) \xrightarrow{b} f(y)}$$

So for each $\psi \in f(f(x))^{-1}(\langle b\rangle\langle a\rangle\top)$ *we have* $\psi(x) = \langle b\rangle\psi(y) = \langle b\rangle(\langle a\rangle\psi(z) \wedge \chi(y)) = \langle b\rangle(\langle a\rangle\top \wedge \chi(y))$ *with* $\chi \in f(y)^{-1}(\langle a\rangle\top)$*. The only P-ruloid (modulo* α*-conversion) with a conclusion* $f(y) \xrightarrow{a} \_$ *is*

$$\frac{y \xrightarrow{b} z \quad z \xrightarrow{a} w}{f(y) \xrightarrow{a} f(z)}$$

So $\chi(y) = \langle b\rangle\chi(z) = \langle b\rangle(\langle a\rangle\chi(w) \wedge \chi'(z)) = \langle b\rangle(\langle a\rangle\top \wedge \chi'(z))$ *for some* $\chi' \in f(z)^{-1}(\top)$*. Since* $\chi'(z) = \top$*,* $\psi(x) = \langle b\rangle(\langle a\rangle\top \wedge \langle b\rangle\langle a\rangle\top)$*.*

By Theorem 3.11 a closed term $f(f(p))$ *can execute a b followed by an a if and only if* $p \xrightarrow{b} p'$ *where* p' *can execute an a and also a b followed by an a.*

3.7 Counterexamples

The following example shows that in Theorem 3.11 it is essential that the TSS is complete. That is, the result would fail if we took the transition relation induced by a TSS to consist of those transitions for which a well-supported proof exists.

Example 3.14 *Let $A = \{a, b\}$ and $P = (\Sigma, R)$, where Σ consists of the constant c and the unary function symbol f, and R is:*

$$\frac{x \overset{a}{\not\longrightarrow}}{f(x) \overset{b}{\longrightarrow} c} \qquad \frac{c \overset{a}{\not\longrightarrow}}{c \overset{a}{\longrightarrow} c}$$

As shown in Example 2.13, this TSS, which is in ready simulation format, is incomplete. In particular, neither $P \vdash_{ws} c \overset{a}{\longrightarrow} p$ for a closed term p nor $P \vdash_{ws} c \overset{a}{\not\longrightarrow}$.

Let us assume that the transition relation induced by this TSS consists of those transitions for which a well-supported proof exists. Then there is no a-transition for c and no b-transition for $f(c)$, so $c \models_P \neg\langle a\rangle\top$ and $f(c) \models_P \neg\langle b\rangle\top$.

The only P-ruloid with a conclusion $f(x) \overset{b}{\longrightarrow} \_$ is $\frac{x \overset{a}{\not\longrightarrow}}{f(x) \overset{b}{\longrightarrow} c}$. Hence Theorem 3.11 would yield $f(c) \models_P \langle b\rangle\top \Leftrightarrow c \models_P \neg\langle a\rangle\top$. Since this is false, Theorem 3.11 would fail with respect to P.

Finally, we argue that Theorem 3.2 and Theorem 3.9 cannot be combined in a straightforward fashion to apply to TSSs that contain both bounded lookahead and negative premises.

Example 3.15 *Let $A = \{a, b\}$ and $P = (\Sigma, R)$, where Σ consists of the constant 0, the unary function symbols $a\_$, $b\_$ and f and the binary function symbol $\_ + \_$, and R is as follows for all $m \in A$:*

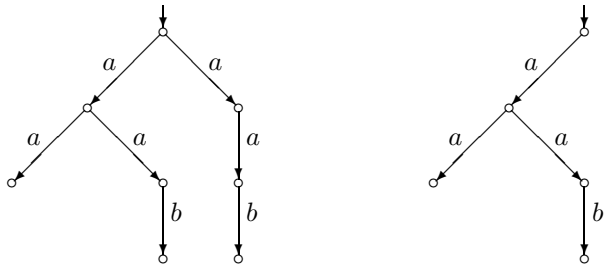
$$\frac{x \overset{m}{\longrightarrow} x'}{x + y \overset{m}{\longrightarrow} x'} \qquad \frac{y \overset{m}{\longrightarrow} y'}{x + y \overset{m}{\longrightarrow} y'} \qquad mx \overset{m}{\longrightarrow} x$$

$$\frac{x \overset{a}{\longrightarrow} y}{f(x) \overset{a}{\longrightarrow} f(y)} \qquad \frac{x \overset{a}{\longrightarrow} y \quad y \overset{b}{\not\longrightarrow}}{f(x) \overset{b}{\longrightarrow} 0}$$

This TSS is complete and in nxyft format. Note that for closed substitutions σ,

$$\begin{array}{lll} & \sigma(f(f(x))) \models \langle b\rangle\top \\ \Leftrightarrow & \sigma(f(x)) \models \langle a\rangle(\neg\langle b\rangle\top) \\ \Leftrightarrow & \sigma(x) \models \langle a\rangle\neg(\langle a\rangle\neg\langle b\rangle\top) \end{array}$$

Suppose that there would exist a notion of P-ruloids, being nxytt rules, such that $P \vdash_{ws} \sigma(f(f(x))) \overset{a}{\longrightarrow} 0$ for σ a closed substitution if and only if there are a P-ruloid $\frac{H}{f(f(x)) \overset{a}{\longrightarrow} u}$ and a closed substitution σ' with $P \vdash_{ws} \sigma'(\alpha)$ for $\alpha \in H$,



$a(a0 + ab0) + aab0 \models \langle a\rangle\neg(\langle a\rangle\neg\langle b\rangle\top)$ $\quad$ $a(a0 + ab0) \not\models \langle a\rangle\neg(\langle a\rangle\neg\langle b\rangle\top)$

Figure 1: Two 2-nested simulation equivalent processes that can be distinguished by the HML formula with nested negations $\langle a\rangle\neg(\langle a\rangle\neg\langle b\rangle\top)$

$\sigma'(x) = \sigma(x)$ *and* $\sigma'(u) = 0$. *In that case we would have, as in Theorem 3.11, that for all closed substitutions* σ*,*

$$\sigma(f(f(x))) \models \langle b\rangle\top \quad \Leftrightarrow \quad \exists\psi \in f(f(x))^{-1}(\langle b\rangle\top) : \sigma(x) \models \psi(x)$$

However, according to Definition 3.10, for each $\psi \in f(f(x))^{-1}(\langle b\rangle\top)$ *and* $y \in V$*,* $\psi(y)$ *does not contain nested negations, whereas the formula* $\langle a\rangle\neg(\langle a\rangle\neg\langle b\rangle\top)$ *above cannot be written as a (possibly infinite) disjunction of HML formulas (as defined in Definition 2.2) without nested negations. The latter follows because the formula* $\langle a\rangle\neg(\langle a\rangle\neg\langle b\rangle\top)$ *distinguishes the two process of Figure 1—yet these processes are* 2-nested simulation equivalent *[14] and 2-nested equivalent processes cannot be distinguished by HML formulas without nested negations [14].*

4 Bisimulation as a Congruence

In [14], Groote & Vaandrager introduced the tyft/tyxt format, and proved that on any LTS specified by a well-founded TSS in tyft/tyxt format, bisimulation equivalence is a congruence, i.e. is preserved by all functions in the signature. Fokkink & van Glabbeek showed in [9] that the well-foundedness condition can be dropped, by constructing for each TSS in tyft/tyxt format an equivalent TSS that consists of pure xyft rules only. In this section we show that the congruence result for TSSs in tyft/tyxt format is a corollary of Theorem 3.11.

Corollary 4.1 *Let* $P = (\Sigma, R)$ *be a TSS in tyft/tyxt format,* $t \in \mathbb{T}(\Sigma)$ *and* σ, σ' *closed substitutions. If* $\sigma(x) \leftrightarrows \sigma'(x)$ *for* $x \in var(t)$*, then* $\sigma(t) \leftrightarrows \sigma'(t)$*.*

Proof: In light of Proposition 2.5, it suffices to show that $\sigma(t)$ and $\sigma(t')$ satisfy the same formulas from $\mathbb{O}$. Let $\sigma(t) \models \varphi \in \mathbb{O}$. By Theorem 3.11 there is a $\psi \in t^{-1}(\varphi)$ such that $\sigma(x) \models \psi(x)$ for $x \in var(t)$. Since $\psi(x) \in \mathbb{O}$ and $\sigma(x) \leftrightarrows \sigma'(x)$ for $x \in var(t)$, it follows by Proposition 2.5 that $\sigma'(x) \models \psi(x)$ for $x \in var(t)$. So by Theorem 3.11, $\sigma'(t) \models \varphi$. By symmetry, $\sigma'(t) \models \varphi \in \mathbb{O}$ implies $\sigma(t) \models \varphi$. □

Likewise, it follows from Theorem 3.11 that on any LTS specified by a complete TSS in ready simulation format, bisimulation equivalence is a congruence. This is a special case of the congruence result from [7, 9] for complete TSSs in ntyft/ntyxt format.

Bloom, Fokkink & van Glabbeek [5] showed how the decomposition method of modal formulas can be used to obtain general congruence formats for a wide range of behavioural semantics. The idea is to take as a starting point the modal characterization of the semantics under scrutiny, this being a set $\mathcal{O}$ of HML formulas such that two processes are semantically equivalent if and only if they make true the same formulas in $\mathcal{O}$. The congruence format for this semantics should guarantee that the decomposition of a formula in $\mathcal{O}$ always produces formulas in $\mathcal{O}$. With the decomposition method in the current paper, we could use this same approach to produce congruence formats that include lookahead. We leave this as future work.

References

[1] H. R. Andersen (1995): *Partial model checking.* In *Proceedings Tenth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, San Diego, California, pp. 398–407.

[2] H.R. Andersen, C. Stirling & G. Winskel (1994): *A compositional proof system for the modal μ-calculus.* In *Proceedings Ninth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, pp. 144–153.

[3] H.R. Andersen & G. Winskel (1992): *Compositional checking of satisfaction. Formal Methods in System Design* 1(4), pp. 323–354.

[4] H. Barringer, R. Kuiper & A. Pnueli (1984): *Now you may compose temporal logic specifications.* In *ACM Symposium on Theory of Computing (STOC '84)*, ACM Press, pp. 51–63.

[5] B. Bloom, W.J. Fokkink & R.J. van Glabbeek (2004): *Precongruence formats for decorated trace semantics. ACM Transactions on Computational Logic* 5(1), pp. 26–78.

[6] B. Bloom, S. Istrail & A.R. Meyer (1995): *Bisimulation can't be traced. Journal of the ACM* 42(1), pp. 232–268.

[7] R. Bol & J.F. Groote (1996): *The meaning of negative premises in transition system specifications. Journal of the ACM* 43(5), pp. 863–914.

[8] S.D. Brookes, C.A.R. Hoare & A.W. Roscoe (1984): *A theory of communicating sequential processes. Journal of the ACM* 31(3), pp. 560–599.

[9] W.J. Fokkink & R.J. van Glabbeek (1996): *Ntyft/ntyxt rules reduce to ntree rules. Information and Computation* 126(1), pp. 1–10.

[10] W.J. Fokkink, R.J. van Glabbeek & P. de Wind (2003): *Compositionality of Hennessy-Milner logic through structural operational semantics.* In A. Lingas & B.J. Nilsson, editors: 14^{th} *International Symposium on Fundamentals of Computation Theory (FCT '03)*, *Lecture Notes in Computer Science* 2751, Springer, pp. 412–422.

[11] R.J. van Glabbeek (2001): *The linear time – branching time spectrum I: The semantics of concrete, sequential processes.* In J.A. Bergstra, A. Ponse & S.A. Smolka, editors: *Handbook of Process Algebra*, chapter 1, Elsevier, pp. 3–99.

[12] R.J. van Glabbeek (2004): *The meaning of negative premises in transition system specifications II. Journal of Logic and Algebraic Programming* 60/61, pp. 229–258.

[13] J.F. Groote (1993): *Transition system specifications with negative premises. Theoretical Computer Science* 118(2), pp. 263–299.

[14] J.F. Groote & F. Vaandrager (1992): *Structured operational semantics and bisimulation as a congruence. Information and Computation* 100(2), pp. 202–260.

[15] D. Harel, D. Kozen & R. Parikh (1982): *Process logic: Expressiveness, decidability, completeness. Journal of Computer and System Sciences* 25(2), pp. 144–170.

[16] M.C.B. Hennessy & R. Milner (1985): *Algebraic laws for nondeterminism and concurrency. Journal of the ACM* 32(1), pp. 137–161.

[17] M.C.B. Hennessy & C. Stirling (1985): *The power of the future perfect in program logics. Information and Control* 67, pp. 23–52.

[18] D. Kozen (1983): *Results on the propositional μ-calculus. Theoretical Computer Science* 27(3), pp. 333–354.

[19] F. Laroussinie & K. G. Larsen (1995): *Compositional model checking of real time systems.* In I. Lee & S. A. Smolka, editors: *CONCUR '95: Concurrency Theory, 6th Conference*, *Lecture Notes in Computer Science* 962, Springer-Verlag, pp. 27–41.

[20] K.G. Larsen (1986): *Context-Dependent Bisimulation between Processes.* PhD thesis, University of Edinburgh, Edinburgh.

[21] K.G. Larsen & X. Liu (1991): *Compositionality through an operational semantics of contexts. Journal of Logic and Computation* 1(6), pp. 761–795.

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

[23] R. Milner (1981): *A modal characterization of observable machine-behaviour.* In E. Astesiano & C. Böhm, editors: *CAAP '81: Trees in Algebra and Programming, 6^{th} Colloquium*, *Lecture Notes in Computer Science* 112, Springer, pp. 25–34.

[24] R. Milner (1983): *Calculi for synchrony and asynchrony. Theoretical Computer Science* 25(3), pp. 267–310.

[25] G. D. Plotkin (2004): *A structural approach to operational semantics. Journal of Logic and Algebraic Programming* 60/61, pp. 17–139. Originally appeared in 1981.

[26] A. Pnueli (1981): *The temporal logic of concurrent programs. Theoretical Computer Science* 13, pp. 45–60.

[27] R. de Simone (1985): *Higher-level synchronising devices in* Meije–*SCCS. Theoretical Computer Science* 37(3), pp. 245–267.

[28] A. Simpson (2004): *Sequent calculi for process verification: Hennessy-Milner logic for an arbitrary GSOS. Journal of Logic and Algebraic Programming* 60/61, pp. 287–322.

[29] C. Stirling (1985): *A proof-theoretic characterization of observational equivalence. Theoretical Computer Science* 39(1), pp. 27–45.

[30] C. Stirling (1985): *A complete modal proof system for a subset of SCCS.* In H. Ehrig, C. Floyd, M. Nivat & J.W. Thatcher, editors: *Mathematical Foundations of Software Development: Proceedings of the Joint Conference on Theory and Practice of Software Development (TAPSOFT), Volume 1: Colloquium on Trees in Algebra and Programming (CAAP '85)*, *Lecture Notes in Computer Science* 185, Springer, pp. 253–266.

[31] C. Stirling (1985): *A complete compositional modal proof system for a subset of CCS.* In W. Brauer, editor: *Automata, Languages and Programming, 12th Colloquium (ICALP '85)*, *Lecture Notes in Computer Science* 194, Springer, pp. 475–486.

[32] C. Stirling (1987): *Modal logics for communicating systems. Theoretical Computer Science* 49(2-3), pp. 311–347.

[33] G. Winskel (1986): *A complete proof system for SCCS with modal assertions. Fundamenta Informaticae* IX, pp. 401–420.

[34] G. Winskel (1990): *On the compositional checking of validity (extended abstract).* In J.C.M. Baeten & J.W. Klop, editors: *CONCUR '90: Theories of Concurrency: Unification and Extension*, *Lecture Notes in Computer Science* 458, Springer, pp. 481–501.