

Executable Specifications in Transition Algebra

A. Riesco (joint work with Daniel Găină and Ionuț Țuțu)

Universidad Complutense de Madrid, Madrid, Spain

Workshop on Logic, Algebra and Category Theory, LAC 2025
Fukuoka, Japan
October 3, 2025

Introduction

- Novel denotational semantics called Transition Algebra (TA), introduced for algebraic specification languages that are executable through rewriting.
- TA supports actions, which are composed from binary relations using union (\cup), composition (\circ), and iteration ($*$).
- TA can provide a rigorous foundation, in logical terms, for a core fragment of the Maude strategy language.
- Maude is a specification language that implements Rewriting Logic.
- The strategy language significantly enhances its expressive power by enabling fine-grained control over the application of transition rules.

Introduction

- TA follows the principles of institution theory, a framework that formalizes the notion of a logical system using four ingredients: signatures, sentences, models, and a satisfaction relation between models and sentences.
- In TA, each signature is equipped with a separate set of plain, unstructured *transition labels*, which are interpreted in models as *atomic transition relations*.
- Here, we adopt a more minimalistic approach.
- We use terms both to denote elements in models and to form atomic transition labels.

Introduction

- It allows us to quantify over atomic transitions using ordinary first-order variables.
- This mechanism serves as a method of encoding second-order quantification (over transition relations) using first-order quantifiers.
- We focus on defining a fragment of TA with good computational properties, making specifications written in this logical framework executable via rewriting or narrowing.
- We introduce a notion of Horn clause in two distinct forms: equational and transitional.
- The framework allows the conditions to include actions that are arbitrarily complex and not necessarily atomic.

Introduction

- We define a system of proof rules for reasoning about clauses in TA.
- The resulting entailment system is both sound and complete.
- Our completeness proof does not rely on the use of a *cut* rule.
- This means that properties of systems specified by clauses in TA can be proved without requiring the discovery of intermediate results.

Introduction - rewriting

- Our rewrite rules are derived from general Horn clauses by imposing restrictions on the occurrences of variables in order to ensure executability.
- Conditions, while not themselves rewrite rules, may include complex compositions of actions and thus require their own operational semantics.
- We proceed in two steps.

Introduction - rewriting

- First, we define a rewriting relation on sets of conditions.
- We interpret these sets of conditions as goals G that are meant to be proved by rewriting.
- Proving a goal amounts to rewriting it to the empty set.
- To prove G we may need to apply other rewrite rules, which may be conditional.
- Unlike term rewriting, for goal rewriting we do not check the conditions of rules straight away; instead, we add them to G so that they are checked at some point during rewriting.
- The purpose is to develop a framework for rewriting that is as general as possible, independent of any particular computational strategy.

Introduction - rewriting

- Second, we define term rewriting using equational rewrite rules and, separately, term transitions using transitional rewrite rules.
- The latter captures the idea of rewriting along a path described by an action, hence giving an operational interpretation to transitions in the system.
- This separation clarifies how terms transition in response to rules and how side conditions are validated through structured rewriting.

Introduction - rewriting

- We establish a completeness result for rewriting, thus ensuring that a goal G follows (semantically) from a set Γ of clauses only if G can be operationally reduced to the empty set through a sequence of rewriting steps.
- This result bridges the gap between the model-theoretic semantics of TA and its operational semantics defined via rewriting.
- To achieve it, the set of rules needs to satisfy two key assumptions: *confluence* of equational rewriting rules and *coherence* of transitional rewrite rules.

Introduction - narrowing

- To define narrowing, we use the same rewrite rules as those used for rewriting.
- The definition of narrowing can be obtained directly from rewriting by replacing matching with unification.
- We consider queries defined as existentially quantified goals, revisit the classical notion of solution to a query, and propose a suitable narrowing relation.
- A query $\exists X \cdot Q$ is satisfiable with respect to a set Γ of clauses, meaning that there exists a substitution θ such that $Q\theta$ follows from Γ , if and only if $\exists X \cdot Q$ can be reduced to a trivial query of the form $\exists X' \cdot \emptyset$ through a sequence of narrowing steps that 'compute' θ .
- The completeness of narrowing holds under confluence, coherence, and also an additional termination requirement for goal rewriting.

Introduction - implementation

- We have implemented a tool prototype for writing TA specifications, reducing terms, checking transitions, and solving queries.
- This was made possible by SpeX, which is both a heterogeneous logical environment and a collection of Maude libraries that facilitate the development of new specification languages.
- Thanks to Maude meta-level, it has been possible to define a dedicated syntax for TA specifications, to parse specifications, transform them into internal Maude representations, and then use those representations to reduce terms and solve queries.

Introduction - implementation

- The close relationship between Rewriting Logic and Transition Algebra makes Maude the most appropriate language for these tasks, as it already provides mechanisms for parsing, matching, and unification.
- Both the sentences and the execution mechanisms of Rewriting Logic and Transition Algebra are different, hence no direct translation is possible.
- For that reason, the tool builds on the features provided by Maude to obtain specially designed resolution procedures for Transition Algebra.

Horn clause in Transition Algebra

- Remember we have $t_1 = t_2$ and $\alpha(t_1, t_2)$.
- A Horn clause is a sentence of the form $\forall X \cdot \varphi$ if H where
 - 1 X is a finite block of variables;
 - 2 H is a finite set of equations and transition rules; and
 - 3 φ is an atomic sentence.

Horn clause in Transition Algebra

We start with atomic sentences.

$$\begin{array}{ll}
 (R) \frac{}{\Gamma \vdash t = t} & (S) \frac{\Gamma \vdash t_1 = t_2}{\Gamma \vdash t_2 = t_1} \\
 (T) \frac{\Gamma \vdash t_1 = t_2 \quad \Gamma \vdash t_2 = t_3}{\Gamma \vdash t_1 = t_3} & \\
 (F) \frac{\Gamma \vdash t_1 = t'_1 \dots \Gamma \vdash t_n = t'_n}{\Gamma \vdash \sigma(t_1, \dots, t_n) = \sigma(t'_1, \dots, t'_n)} & (L) \frac{\Gamma \vdash t_i = t'_i \text{ for both } i \in \{1, 2\} \quad \Gamma \vdash \tau = \tau' \quad \Gamma \vdash \tau(t_1, t_2)}{\Gamma \vdash \tau'(t'_1, t'_2)}
 \end{array}$$

These rules are sound.

Horn clause in Transition Algebra

We add transition relations and quantifier-free clauses.

$$(Comp) \frac{\Gamma \vdash \alpha_1(t_1, t_2) \quad \Gamma \vdash \alpha_2(t_2, t_3)}{\Gamma \vdash (\alpha_1 \circ \alpha_2)(t_1, t_3)}$$

$$(Union) \frac{\Gamma \vdash \alpha_i(t_1, t_2)}{\Gamma \vdash (\alpha_1 \cup \alpha_2)(t_1, t_2)}$$

$$(Eq) \frac{\Gamma \vdash t_1 = t_2}{\Gamma \vdash \alpha^0(t_1, t_2)}$$

$$(Star) \frac{\Gamma \vdash \alpha^n(t_1, t_2)}{\Gamma \vdash \alpha^*(t_1, t_2)}$$

$$(MP) \frac{\Gamma \vdash \varphi \text{ if } H \quad \Gamma \vdash \phi \text{ for all } \phi \in H}{\Gamma \vdash \varphi}$$

$$(Subst) \frac{\Gamma \vdash \forall X \cdot \gamma}{\Gamma \vdash \gamma\theta} \text{ for all } \theta : X \rightarrow \emptyset$$

These* rules are sound.

Horn clause in Transition Algebra

We add transition rules and quantified clauses.

$$(Comp_I) \frac{\Gamma \cup \{a_1(t_1, x), a_2(x, t_2)\} \vdash_{\Sigma[z]} \gamma}{\Gamma \cup \{(a_1 ; a_2)(t_1, t_2)\} \vdash \gamma}$$

$$(Union_I) \frac{\Gamma \cup \{a_i(t_1, t_2)\} \vdash \gamma \text{ for all } i \in \{1, 2\}}{\Gamma \cup \{(a_1 \cup a_2)(t_1, t_2)\} \vdash \gamma}$$

$$(Star_I) \frac{\Gamma \cup \{a^n(t_1, t_2)\} \vdash \gamma \text{ for all } n \in \mathbb{N}}{\Gamma \cup \{a^*(t_1, t_2)\} \vdash \gamma}$$

$$(Imp) \frac{\Gamma \cup H \vdash \varphi}{\Gamma \vdash \varphi \text{ if } H}$$

$$(UQ_E) \frac{\Gamma \vdash_{\Sigma} \forall X \cdot \gamma}{\Gamma \vdash_{\Sigma[X]} \gamma}$$

$$(UQ_I) \frac{\Gamma \vdash_{\Sigma[X]} \gamma}{\Gamma \vdash_{\Sigma} \forall X \cdot \gamma}$$

Horn clause in Transition Algebra

Lemma (Soundness)

The proof rules defined in the tables above are sound. That is, the least entailment system closed under the proof rules defined in the tables above is sound.

Theorem (Completeness)

Let \vdash be any sound entailment system closed under the proof rules defined in the tables above. For all sets of clauses Γ and all clauses γ defined over the same signature, we have $\Gamma \models \gamma$ iff $\Gamma \vdash \gamma$.

Operational semantics - rewriting

- We consider two kinds of rewrite rules to ‘execute’ specifications.
- An *equational rewrite rule* is a Horn clause of the form $\forall X \cdot (\ell = r) \text{ if } H$ such that ℓ is not a variable and $X = \text{var}(\ell) \cup \text{var}(H)$.
- A *transitional rewrite rule* is a Horn clause of the form $\forall X \cdot \tau(\ell, r) \text{ if } H$ such that ℓ is not a variable and $X = \text{var}(\tau) \cup \text{var}(\ell) \cup \text{var}(H)$.
- Whenever we refer to a *term rewriting system* in Transition Algebra we mean a pair (Σ, Γ) consisting of a signature Σ and a set $\Gamma \subseteq \text{Sen}(\Sigma)$ of equational and transitional rewrite rules.

Operational semantics - rewriting

Definition (Rewriting step)

Let (Σ, Γ) be a term rewriting system. The *one-step-rewriting* relation on ground equations and transitions is defined as follows:

- EQ** $C[t] \xrightarrow{\text{EQ}} C[r\theta] \cup H\theta$ whenever $t = \ell\theta$ for some rule $\forall Y \cdot (\ell = r)$ if $H \in \Gamma$ and substitution $\theta : Y \rightarrow \emptyset$;
- =** $\{t = t\} \cup G \xrightarrow{=} G$;
- TR** $\{\tau(t_1, t_2)\} \cup G \xrightarrow{\text{TR}} \{r\theta = t_2\} \cup H\theta \cup G$ whenever $\tau = \kappa\theta$ and $t_1 = \ell\theta$ for some $\forall Y \cdot \kappa(\ell, r)$ if $H \in \Gamma$ and $\theta : Y \rightarrow \emptyset$;
- ;** $\{(a_1 ; a_2)(t_1, t_2)\} \cup G \xrightarrow{;} \{a_1(t_1, t), a_2(t, t_2)\} \cup G$, for any Σ -term t ;
- \cup** $\{(a_1 \cup a_2)(t_1, t_2)\} \cup G \xrightarrow{\cup} \{a_i(t_1, t_2)\} \cup G$, for any $i \in \{1, 2\}$;
- \circ** $\{a^*(t_1, t_2)\} \cup G \xrightarrow{\circ} \{a^n(t_1, t_2)\} \cup G$, for any $n \in \mathbb{N}$.

Operational semantics - rewriting

Definition (Rewriting)

The general *rewriting* relation \rightsquigarrow^* on Transition Algebra goals is the reflexive and transitive closure of the one-step-rewriting relation. We define it by $\rightsquigarrow^* = \bigcup \{ \rightsquigarrow^n \mid n \in \mathbb{N} \}$, where:

$G \rightsquigarrow^0 G$, and

if $G_1 \rightsquigarrow^{lb} G_2$ and $G_2 \rightsquigarrow^n G_3$, then $G_1 \rightsquigarrow^{n+1} G_3$, for each label $lb \in \{EQ, =, TR, \circ, \cup, \odot\}$.

Operational semantics - rewriting

- So far, we have defined rewriting on sets of ground equations and transitions.
- To extend it to equations and transitions with variables from some given block X , we treat the variables in X as new constants that we add to the underlying signature.
- Concretely, for any term rewriting system (Σ, Γ) and block X of variables, rewriting sets of equations and transitions with variables from X amounts to applying rewriting over $(\Sigma[X], \Gamma)$.

Operational semantics - rewriting

This notion of goal rewriting is sound in the following sense: in the presence of the rewriting rules (Horn clauses) provided by Γ , the outcome of a rewrite is always stronger than the original goal.

Theorem (Soundness of rewriting)

Let (Σ, Γ) be a term rewriting system. For any goals $G, G' \subseteq \text{Sen}(\Sigma)$,

$$G \rightsquigarrow^* G' \text{ implies } \Gamma \cup G' \models G.$$

Corollary

For any term rewriting system (Σ, Γ) and goal $G \subseteq \text{Sen}(\Sigma)$, $G \rightsquigarrow^ \emptyset$ implies $\Gamma \models G$.*

Operational semantics - narrowing

- Next, we look for solutions to *queries*, which are defined as existentially quantified goals and are usually denoted $\exists X \cdot Q$, where X is a block of variables and Q is a goal with variables from X .
- As in equational logic programming, a *solution* to $\exists X \cdot Q$ is a substitution $\theta: X \rightarrow X'$ such that $\Gamma \models \forall X' \cdot Q\theta$.
- We solve queries by narrowing, which is similar to rewriting, but employs most general unifiers instead of matching substitutions in order to compute 'values' for the variables in X such that the equalities and transitions in Q hold true.

Operational semantics - narrowing

Definition (Narrowing step)

Let (Σ, Γ) be a term rewriting system. The *one-step-narrowing* relation on queries, indexed by so-called *computed substitutions*, is defined as follows:

EQ $\exists X \cdot C[t] \xrightarrow{\text{EQ}}_{\psi_X} \exists X' \cdot (C[r] \cup H)\psi$ when t is not a variable,

$\psi = \text{mgu}\{t \doteq \ell\} : X \uplus Y \rightarrow X'$ for some rule $\forall Y \cdot (\ell = r)$ if $H \in \Gamma$, and $\psi_X : X \rightarrow X'$ is the restriction of ψ to X ;

= $\exists X \cdot \{t_1 = t_2\} \cup Q \xrightarrow{=}_{\psi} \exists X' \cdot Q\psi$ when $\psi = \text{mgu}\{t_1 \doteq t_2\} : X \rightarrow X'$;

TR $\exists X \cdot \{\tau(t_1, t_2)\} \cup Q \xrightarrow{\text{TR}}_{\psi_X} \exists X' \cdot (\{r = t_2\} \cup H \cup Q)\psi$ when



$\psi = \text{mgu}\{\tau \doteq \kappa, t_1 \doteq \ell\} : X \uplus Y \rightarrow X'$ for some rule $\forall Y \cdot \kappa(\ell, r)$ if $H \in \Gamma$ and $\psi_X : X \rightarrow X'$ is the restriction of ψ to X ;

; $\exists X \cdot \{(a_1 ; a_2)(t_1, t_2)\} \cup Q \xrightarrow{;}_{\iota_x} \exists X \cup \{x\} \cdot \{a_1(t_1, x), a_2(x, t_2)\} \cup Q$, for any new variable x , distinct from those in X , and for the obvious inclusion $\iota_x : X \hookrightarrow X \cup \{x\}$;

Operational semantics - narrowing

Definition (Narrowing step)

Let (Σ, Γ) be a term rewriting system. The *one-step-narrowing* relation on queries, indexed by so-called *computed substitutions*, is defined as follows:

- 
 $\exists X \cdot \{(a_1 \cup a_2)(t_1, t_2)\} \cup Q \xrightarrow{\cup}_{1_X} \exists X \cdot \{a_i(t_1, t_2)\} \cup Q$, for any $i \in \{1, 2\}$, where $1_X: X \rightarrow X$ is the identity; and
- 
 $\exists X \cdot \{a^*(t_1, t_2)\} \cup Q \xrightarrow{\circ}_{1_X} \exists X \cdot \{a^n(t_1, t_2)\} \cup Q$, for any $n \in \mathbb{N}$, where $1_X: X \rightarrow X$ is the identity.

Operational semantics - narrowing

The narrowing relation corresponds to the reflexive and transitive closure of the one-step-narrowing relation.

Definition (Narrowing)

The general *narrowing* relation is defined by $\xrightarrow{*}_\psi = \bigcup \{ \xrightarrow{n}_\psi \mid n \in \mathbb{N} \}$, where:

- 1 $\exists X \cdot Q \xrightarrow{0}_{1_X} \exists X \cdot Q$, and
- 2 if $\exists X_1 \cdot Q_1 \xrightarrow{lb}_{\psi_1} \exists X_2 \cdot Q_2$ and $\exists X_2 \cdot Q_2 \xrightarrow{n}_{\psi_2} \exists X_2 \cdot Q_3$, then
 $\exists X_1 \cdot Q_1 \xrightarrow{n+1}_{\psi_1 \circ \psi_2} \exists X_3 \cdot Q_3$, for each label $lb \in \{EQ, =, TR, \circ, \cup, \cap\}$.

Operational semantics - narrowing

The following result establishes our first link between narrowing and rewriting.

Lemma (Reduction Lemma)

If $\exists X \cdot Q \xrightarrow{n}_{\psi} \exists X' \cdot Q'$, then $Q\psi \rightsquigarrow Q'$.

One important consequence of Reduction Lemma is the soundness of narrowing.

Theorem (Soundness of narrowing)

Let (Σ, Γ) be a term rewriting system. If $\exists X \cdot Q \xrightarrow{}_{\psi} \exists X' \cdot Q'$, then for any solution $\theta: X' \rightarrow X''$ to $\exists X' \cdot Q'$, the substitution $\psi \circ \theta$ is a solution to $\exists X \cdot Q$.*

Operational semantics - term rewriting

Definition (Term rewriting)

Let (Σ, Γ) be a term rewriting system. We overload the notation used for rewriting goals and define a *one-step-rewriting* relation on Σ -terms as follows: for any context c and term t , $c[t] \xrightarrow{\text{EQ}} c[r\theta]$ if there exist an equational rule $\forall Y \cdot (\ell = r)$ if $H \in \Gamma$ and a substitution $\theta: Y \rightarrow \emptyset$ such that $t = \ell\theta$ and $H\theta \rightsquigarrow^* \emptyset$.




The family of *n-step rewriting* relations $\{\xrightarrow{n} \subseteq T_\Sigma \times T_\Sigma \mid n \in \mathbb{N}\}$ is defined inductively as follows:

- 1 $t \xrightarrow{0} t$;
- 2 if $t_1 \xrightarrow{\text{EQ}} t_2$ and $t_2 \xrightarrow{n} t_3$, then $t_1 \xrightarrow{n+1} t_3$.

We write $t \xrightarrow{*} t'$ whenever $t \xrightarrow{n} t'$ for some $n \in \mathbb{N}$.

Operational semantics - term rewriting

Term rewriting extends canonically from terms to actions:

-  $a_1 \circ a_2 \xrightarrow{*} a'_1 \circ a'_2$ when $a_1 \xrightarrow{*} a'_1$ and $a_2 \xrightarrow{*} a'_2$;
-  $a_1 \cup a_2 \xrightarrow{*} a'_1 \cup a'_2$ when $a_1 \xrightarrow{*} a'_1$ and $a_2 \xrightarrow{*} a'_2$; and
-  $a^* \xrightarrow{*} b^*$ when $a \xrightarrow{*} b$.

Operational semantics - term rewriting

Term rewriting together with action rewriting allows us to define a transition relation on terms.

Definition (Term transition)

Let (Σ, Γ) be a term rewriting system. The relation \xrightarrow{a} on Σ -terms is defined by induction on the structure of a :

TR $t_1 \xrightarrow{\tau} t_2$ if there exist a rewrite rule $\forall Y \cdot \kappa(\ell, r)$ if $H \in \Gamma$ and a substitution $\theta : Y \rightarrow \emptyset$ such that

- 1 $t_1 \xrightarrow{*} \ell\theta$,
- 2 $\tau \xrightarrow{*} \kappa\theta$,
- 3 $r\theta \xrightarrow{*} t_2$, and
- 4 $H\theta \xrightarrow{*} \emptyset$;

; $t_1 \xrightarrow{a_1 ; a_2} t_2$ if $t_1 \xrightarrow{a_1} t$ and $t \xrightarrow{a_2} t_2$ for some term $t \in T_\Sigma$;

U $t_1 \xrightarrow{a_1 \cup a_2} t_2$ if $t_1 \xrightarrow{a_i} t$ for some $i \in \{1, 2\}$; and

∘ $t_1 \xrightarrow{a^*} t_2$ if $t_1 \xrightarrow{a^n} t_2$ for some $n \in \mathbb{N}$.

Operational semantics - term rewriting

Lemma

Let (Σ, Γ) be a term rewriting system. For all Σ -actions a, b and all Σ -terms t_1, t_2 ,

if $a \xrightarrow{*} b$ and $t_1 \xrightarrow{b} t_2$, then $t_1 \xrightarrow{a} t_2$.

Lemma

Let (Σ, Γ) be a term rewriting system.

- ① If $t_1 \xrightarrow{*} u \xleftarrow{*} t_2$, then $\{t_1 = t_2\} \rightsquigarrow \emptyset$.
- ② If $t_1 \xrightarrow{a} u \xleftarrow{*} t_2$, then $\{a(t_1, t_2)\} \rightsquigarrow \emptyset$.

Operational semantics - term rewriting

Corollary

Let (Σ, Γ) be a term rewriting system.

- Term rewriting is sound, meaning that $t_1 \xrightarrow{*} t_2$ implies $\Gamma \models t_1 = t_2$.
- Term transition is sound, meaning that $t_1 \xrightarrow{a} t_2$ implies $\Gamma \models \mathbf{a}(t_1, t_2)$.

Operational semantics - term rewriting

Proposition

Let (Σ, Γ) be a ground confluent term rewriting system.

- 1 If $\{t_1 = t_2\} \rightsquigarrow^* \emptyset$, then $t_1 \xrightarrow{*} u \xleftarrow{*} t_2$ for some Σ -term u .
- 2 Moreover, if (Σ, Γ) is coherent and $\{a(t_1, t_2)\} \rightsquigarrow^* \emptyset$, then $t_1 \xrightarrow{a} u \xleftarrow{*} t_2$ for some Σ -term u .

Completeness of operational semantics - rewriting

All completeness results are based on confluence and coherence assumptions.

Theorem (Completeness of rewriting)

Let (Σ, Γ) be a ground confluent and coherent term rewriting system. For all goals $G \subseteq \text{Sen}(\Sigma)$,

$$G \rightsquigarrow^* \emptyset \quad \text{if} \quad \Gamma \models G.$$

Completeness of operational semantics - narrowing

Definition (Strict goal rewriting)

We say that a goal G *rewrites strictly* to G' , and write $G \rightsquigarrow^{*!} G'$, when there exists a sequence of rewriting steps $G \rightsquigarrow G_1 \rightsquigarrow G_2 \rightsquigarrow \dots \rightsquigarrow G'$ where all the occurrences of the ' \circledast ' step are instances of the following stricter decomposition of actions:

$$\boxed{\circledast!} \quad \{(a_1 \circledast a_2)(t_1, t_2)\} \cup G \rightsquigarrow \{a_1(t_1, t), a_2(t, t_2)\} \cup G, \text{ for any normal } \Sigma\text{-term } t.$$

That is, $\rightsquigarrow^{*!} = \bigcup \{ \rightsquigarrow^{n!} \mid n \in \mathbb{N} \}$, where:

- ① $G \rightsquigarrow^{0!} G$, and
- ② if $G_1 \rightsquigarrow^{lb} G_2$ and $G_2 \rightsquigarrow^{n!} G_3$, then $G_1 \rightsquigarrow^{n+1!} G_3$, for each label $lb \in \{EQ, =, TR, \circledast!, \cup, \circlearrowleft\}$.

Completeness of operational semantics - narrowing

Proposition

Let (Σ, Γ) be a canonical and coherent term rewriting system, and let $\text{nf}(t_1)$ and $\text{nf}(t_2)$ be the (unique) normal forms of terms t_1 and t_2 , respectively. For any action α , if $\{\alpha(t_1, t_2)\} \rightsquigarrow^* \emptyset$, then

- ① $\{\alpha(t_1, \text{nf}(t_2))\} \rightsquigarrow^* \emptyset$, and
- ② $\{\alpha(\text{nf}(t_1), t_2)\} \rightsquigarrow^* \emptyset$.

Completeness of operational semantics - narrowing

Lemma (Lifting Lemma)

Let (Σ, Γ) be a term rewriting system such that:

- ① for each equational rule $\forall Y \cdot (\ell = r)$ if $H \in \Gamma$, $Y = \text{var}(l)$, and
- ② for each transition rule $\forall Y \cdot \kappa(\ell, r)$ if $H \in \Gamma$, $Y = \text{var}(\kappa) \cup \text{var}(\ell)$.

For every query $\exists X \cdot Q$, normal substitution $\theta: X \rightarrow Z$, and goal G over $\Sigma[Z]$ such that $Q\theta \xrightarrow{n!} G$, there exist a narrowing derivation $\exists X \cdot Q \xrightarrow{n \gg \psi} \exists X' \cdot Q'$ and a normal substitution $\delta: X' \rightarrow Z$ such that $Q'\delta = G$, and $\psi \circ \delta = \theta$.

$$\begin{array}{ccc}
 \exists X \cdot Q & \xrightarrow{n \gg \psi} & \exists X' \cdot Q' \\
 \theta \downarrow & & \downarrow \delta \\
 Q\theta & \xrightarrow{n!} & G
 \end{array}$$

Completeness of operational semantics - narrowing

Theorem

Let (Σ, Γ) be a canonical and coherent term rewriting system such that:

- ① for each equational rule $\forall Y \cdot (\ell = r)$ if $H \in \Gamma$, $Y = \text{var}(\ell)$, and
- ② for each transition rule $\forall Y \cdot \kappa(\ell, r)$ if $H \in \Gamma$, $Y = \text{var}(\kappa) \cup \text{var}(\ell)$.

For every normal solution $\theta: X \rightarrow Z$ to a query $\exists X \cdot Q$ there exists a narrowing derivation $\exists X \cdot Q \xrightarrow{*}_{\psi} \exists X' \cdot \emptyset$ and a substitution $\delta: X' \rightarrow Z$ such that $\theta = \psi \circ \delta$.

Corollary (Completeness of narrowing)

Under the conditions above,

$$\exists X \cdot Q \xrightarrow{*}_{\psi} \exists X' \cdot \emptyset \quad \text{if} \quad \Gamma \models \exists X \cdot Q.$$

DEMO

`https://github.com/Transition-Algebra/TATP`

TATP - Implementation notes (rewriting)

- We can take advantage of the meta-level capabilities of Maude for designing a resolution procedure for reductions.
- Given a term t , we look for an equation $\ell = r$ if H , such that ℓ matches t , obtaining a substitution θ and a context c .
- Then, a new substitution θ' extending θ is required for satisfying the condition H ; θ' will be used for instantiating r , replacing ℓ in the context c .
- The process is repeated until no equations can be applied to the current term.
- The condition H may contain both equalities and transitions.
- For equalities, the current substitution is applied to both sides of the equation and they are reduced independently as described.
- Once they cannot be further reduced, equality modulo axioms is checked.

TATP - Implementation notes (rewriting)

- For transitions, we also apply the current substitution, but the process is slightly different to the one before.
- First, the matching is twofold: we need the term to match the left-hand side and the actions to match the labels.
- Then, the implementation includes built-in transitions implementing the rules for composition, choice, and iteration, which are used when required.
- Finally, the matching with the right-hand side might extend the substitution.

TATP - Implementation notes (rewriting)

- Then, θ' is computed by using backtracking: for each transition in the condition, as many child nodes can be generated as there are possible substitutions.
- It is important to note that, because we work modulo equational axioms such as associativity and commutativity, several different substitutions can be generated when matching a term and a pattern.
- The number of substitutions generated by a transition might be infinite.
- For this reason, the traversal of the tree follows a depth-first strategy and each substitution is computed lazily, only if required.
- Also note that equalities must be recomputed in general when the substitution changes.

TATP - Implementation notes (rewriting)

- Termination is ensured by supporting two bounds: in the number of steps (including both reductions and transitions) and in the number of conditions.
- The bounds are considered differently.
- The number of steps is computed globally, so each application of an equation or a transition reduces the bound
- The number of conditions is used locally for each reduction step and restarted again for the next one.
- Our implementation stores the terms already tried to reduce in the conditions, avoiding loops that often occur when dealing with some operators, in particular iteration.

TATP - Implementation notes (narrowing)

- For solving queries, we adapt a general logic-programming procedure that is already integrated into SpeX and matches closely our definition of narrowing.
- This procedure relies on the Maude strategy language and involves the rewriting of three kinds of 'configurations': *initial*, *open*, and *solved*.
- An initial configuration is a pair consisting of a specification (set of Horn clauses) and a query that is meant to be checked.
- Optionally, for `echeck`, an initial configuration may also provide a list of (sets of) clauses (identified by labels) to be used in subsequent narrowing steps.

TATP - Implementation notes (narrowing)

- An open configuration includes, besides the specification and a query, information about the partial solution computed up to that point.
- This means that, for any chain of narrowing steps $\exists X_0 \cdot Q_0 \rightarrow_{\psi_1} \exists X_1 \cdot \emptyset_1 \rightarrow_{\psi_2} \exists X_2 \cdot \emptyset_2 \rightarrow_{\psi_3} \dots$ we have a corresponding chain of configuration rewrites such that, for each index n , the n th open configuration includes an encoding of the substitution $\psi_1 \circ \psi_2 \circ \dots \circ \psi_n$.
- Open configurations may also carry lists of (sets of) clauses for `eccheck` and information about the variables in use.
- The latter information is passed to built-in Maude functions for computing unifiers.
- Finally, a solved configuration is a configuration whose query is \top .

TATP - Implementation notes (narrowing)

- The rewriting of configurations is controlled using Maude strategies. Every change of configurations consists of several sub-steps:
 - 1 first, we non-deterministically select a context or formula within the current query where narrowing is applied;
 - 2 then we select (also non-deterministically) a suitable clause,
 - 3 we compute all possible unifiers (which may not be unique if the computations are performed modulo associativity and commutativity),
 - 4 apply substitutions,
 - 5 compile new partial solutions, and
 - 6 update the information on the variable names that are currently in use and on the list of (sets of) clauses meant to guide the following narrowing steps.

TATP - Implementation notes (narrowing)

- Similarly to term rewriting, termination can be enforced by specifying a bound on the number of narrowing steps that can be applied.
- Still, the search space for solved configurations can easily become infeasibly large even for low bounds.
- Because of this, the explicit search (implemented for `echeck`) is often more convenient as it drastically limits the extent of the search space by specifying which clauses are admissible at each step.
- Its downside is that the search procedure is no longer fully automatic, and completeness cannot be guaranteed.
- To counterbalance this limitation, if no solved configuration is found (given a sequence HCL of Horn-clause labels), then the explicit search returns a final list of open configurations that may be rewritten to a solved one (by modifying or extending HCL).

Conclusions

- We have examined Transition Algebra, an extension of many-sorted first-order logic that includes features from dynamic logic, where actions are used as labels of transitions, thus allowing us to capture and analyse complex behaviours.
- This logic is useful for modelling and studying systems where the actions are as important as the system states upon which they produce an effect.
- Unlike previous definitions, and for greater flexibility in the use of actions, we have extended (and simplified) the framework in order to support arbitrary terms as actions.
- We have devised a sound and complete set of proof rules for reasoning about Horn clauses.
- We have developed an operational semantics for Transition Algebra based on term rewriting, transition checking, and narrowing.
- We have shown that rewriting and narrowing are both sound and complete, thus advocating for Transition Algebra as an appropriate framework for system verification.

Conclusions

- The features above provide advantages from both theoretical and practical points of view.
- On the one hand, systems can be more easily and accurately described than with conventional equational logics.
- Furthermore, Transition Algebra allows for integrating specification and verification organically within a single framework, which is useful especially when considering properties that cannot be easily expressed equationally.
- On the other hand, its simple but powerful operational semantics is very well suited for axiomatization.
- We have implemented a Maude prototype that supports specifications in Transition Algebra and solving queries on them.

Ongoing work

- On the theoretical side, we want to explore additional action operators, such as tests and parallel composition.
- We aim to continue the study of narrowing.
- Though sound and complete, in its current form, narrowing is still rather inefficient from a computational perspective.
- We intend to look further into different forms of narrowing, as can be found in the equational-logic literature.
- Moreover, we aim to explore the use of explicit narrowing as a useful tool in interactive theorem proving.
- On the practical side, we are interested in continuing the development of the tool.
- We also plan to specify and analyse a variety of systems to assess the approach and to design strategies to ease the verification process.