CS 524
Lecture 11
José Meseguer

Rewriting Logic Dynamics: Rewrite Theories and their Computations

1. Rewrite theories

In Lecture 10 a full answer was given to the *statics* question:

what is a concurrent state?

namely: an element $[u]_{\equiv_E} \in T_{\Sigma/E}$ of the *initial algebra* of an equational theory (Σ, E) such that $T_{\Sigma/E}$ is a *computable* algebra. To do full justice to this answer, Σ should be order-sorted or even a membership equational logic signature; but to keep the notation simpler the 1-sorted case is treated: everything generalizes naturally to the order-sorted and MEL cases.

Next comes the *dynamics* question:

> What is a *local* *atomic* *transition* in a concurrent system [whose states have the form $[u]_{\equiv_E} \in T_{\Sigma/E}$]?

where by *local* is meant that *it need not be global*, i.e., it may change *some part* of the

concurrent state [leaving other parts unchanged] as opposed to the entire [global] state. By atomic, of course it is meant that such a local transition is not decomposable into simpler, smaller transitions: it happens all at once.

The answer given in rewriting logic to this question is:

Such a local concurrent transition is the instance for $[u_1], \ldots, [u_n] \in T_{\Sigma/E}$ (Notation: $[u]_{=_E}$ is abbreviated to $[u]$) of a parametric transition of the form:

$$l : t(x_1, \ldots, x_n) \longrightarrow t'(x_1, \ldots, x_n)$$

where l is a label and $t, t' \in T_\Sigma(X)$, with x_n the biggest variable appearing in those of t or t'.

For each $[u_1], \ldots, [u_n]$ this rule defines the local atomic transition

$$l(u_1, \ldots, u_n) : [t(u_1, \ldots, u_n)] \longrightarrow [t'(u'_1, \ldots, u'_n)]$$

changing the state fragment $[t(u_1, \ldots, u_n)]$ (which may be just part of a bigger state) into the state fragment

$[t'(u_1,..,u_n)]$. (Notation: $t(u_1,..,u_n) =_{def} t\{x_1 \mapsto u_1, .., x_n \mapsto u_n\}$).

An obvious example is CL, with states in $T_{\Sigma_{CL}}$ and the K-red, S-red and I-red rules.

But note that where t, t' are ground terms $t, t' \in T_\Sigma$ is not excluded, since $T_\Sigma \subseteq T_\Sigma(X)$. Therefore, any automaton transition $a : q \to q'$, or any Petri net transition $\ell : m \to m'$ are instances of this general notion, as degenerate [unparametric] cases.

We called $\ell : t \to t'$ a [labeled] (Σ-) rewrite rule.

Now, the million-dollar question:

> How can general concurrent systems be naturally specified?

Has an obvious answer:

By rewrite theories specifying the concurrent system states as concurrent data structures $[u] \in T_{\Sigma/E}$ for some (Σ, E), and its parametric local concurrent transitions as rewrite rules $\ell : t \to t'$. In more detail:

Definition [Rewrite theory]. A rewrite theory is a triple $\mathcal{R} = (\Sigma, E, R)$ where: (i) (Σ, E) is an equational theory, and R is a set of labeled Σ-rewrite rules.

The set of labels of $\mathcal{R}$, $L_{\mathcal{R}}$ is, by definition,

$$L_{\mathcal{R}} = \{ l \mid \exists (l : t \to t') \in R \}$$

we always assume in what follows that whenever $(l : t \to t'), (l : u \to u') \in R$, then $t = t'$ and $u = u'$ [rule names are globally unique].

A rewrite theory $\mathcal{R} = (\Sigma, E, R)$ does obviously specify a concurrent system whose concurrent states are naturally modeled as concurrent data structures $[u] \in T_{\Sigma/E}$, and whose local atomic concurrent transitions have the form:

$$l(u_1, .., u_n) : [t(u_1, .., u_n)] \longrightarrow [t'(u_1, .., u_n)]$$

for some $[u_1], .., [u_n] \in T_{\Sigma/E}$, $\big(l : t(x_1, .., x_n) \to t'(x_1, .., x_n)\big) \in R$.

But then, the obvious next question is:

> What are all the finite concurrent computations of the concurrent system specified by $\mathcal{R}$?

For anybody familiar with the examples of rewrite theories:

1. A <u>finite automaton</u> $\mathcal{A} = (Q, L, \rightarrow_{\mathcal{A}})$, as the rewrite theory $\mathcal{R}_{\mathcal{A}} = (\Sigma_Q, \emptyset, \rightarrow_{\mathcal{A}})$, where $\Sigma_{Q,0} = Q$, and $\Sigma_{Q,n} = \emptyset$ otherwise

2. A <u>Petri net</u> $\mathcal{N} = (M(P), L, \rightarrow_{\mathcal{N}})$, as the rewrite theory $\mathcal{R}_{\mathcal{N}} = (\Sigma_P, ACU, \rightarrow_{\mathcal{N}})$, where $\Sigma_{P,0} = P \cup \{null\}$ $\Sigma_{P,2} = \{--\}$, $\Sigma_{P,n} = \emptyset$ otherwise, and

$$ACU = \left\{ \begin{array}{l} (x_1\, x_2)\, x_3 = x_1\, (x_2\, x_3), \\ x_1\, x_2 = x_2\, x_1 \\ x_1\, null = x_n \end{array} \right\}$$

3. <u>Combinatory logic</u>, as the rewrite theory $\mathcal{R}_{CL} = (\Sigma_{CL}, \emptyset, R_{CL})$, where R_{CL} are the K-red, S-red, and I-red reduction rules

The answer to the above question is [almost] obvious, since it is just the <u>natural generalization</u> of the <u>domain-specific answers</u> given for automata, Petri nets an combinatory logic, and is given in <u>two steps</u>:

Step 1: The logic $\mathcal{L}(\mathcal{R})$ specifies all concurrent computations

For $\mathcal{R}=(\Sigma, E, R)$ a rewrite theory, the computational logic $\mathcal{L}(\mathcal{R})$ of $\mathcal{R}$ defines the computation specifications of $\mathcal{R}$ as the labeled arrows $[u] \xrightarrow{\alpha} [v]$ provable by the following inference system:

Reflexivity/Idle

$$\frac{}{[t] \xrightarrow{t} [t]} \quad \text{if } [t] \in T_{\Sigma/E}$$

Congruence

$$\frac{[u_1] \xrightarrow{\alpha_1} [v_1] \quad \cdots \quad [u_n] \xrightarrow{\alpha_n} [v_n]}{[f(u_1,..,u_n)] \xrightarrow{f(\alpha_1,..,\alpha_n)} [f(v_1,..,v_n)]} \begin{cases} \text{if } n \geq 1 \\ f \in \Sigma_n \end{cases}$$

Replacement

$$\frac{[u_1] \xrightarrow{\alpha_1} [v_1] \quad \cdots \quad [u_n] \xrightarrow{\alpha_n} [v_n]}{[t(u_1,..,u_n)] \xrightarrow{l(\alpha_1,..,\alpha_n)} [t'(v_1,..,v_n)]} \begin{cases} \text{if} \\ l : t(x_1,..x_n) \to t'(x_1,..,x_n) \\ \in R \end{cases}$$

Transitivity

$$\frac{[u] \xrightarrow{\alpha} [v] \quad [v] \xrightarrow{\beta} [w]}{[u] \xrightarrow{\alpha;\beta} [w]}$$

Notation $[u] \xrightarrow{\alpha} [v]$ provable in $\mathcal{L}(\mathcal{R})$ is denoted as:

$$\mathcal{L}(\mathcal{R}) \vdash [u] \xrightarrow{\alpha} [v].$$

$\mathcal{L}(\mathcal{R})$ defines the labeled graph $Spec_{\mathcal{R}} = (T_{\Sigma/E}, PT_{\mathcal{R}}, \xrightarrow[Spec(\mathcal{R})]{})$

of its computation specifications, where, by definition,

1. The set $PT_{\mathcal{R}}$ of proof terms of $\mathcal{R}$ labeling the arrows is:

$$PT_{\mathcal{R}} = \{\alpha \mid \mathcal{L}(\mathcal{R}) \vdash [u] \xrightarrow{\alpha} [v] \text{ for some } [u],[v] \in T_{\Sigma/E}\}$$

2. $$([u] \xrightarrow{\alpha} [v]) \in \xrightarrow[Spec(\mathcal{R})]{} \Leftrightarrow \mathcal{L}(\mathcal{R}) \vdash [u] \xrightarrow{\alpha} [v]$$

Furthermore $Spec_{\mathcal{R}}$ is a Σ-algebra on its nodes $(T_{\Sigma/E})$, labels $PT_{\mathcal{R}}$, where $\forall n \in \mathbb{N}, \forall f \in \Sigma_n$

$$f_{PT_{\mathcal{R}}} =_{def} \lambda(\alpha_1,..,\alpha_n) \in PT_{\mathcal{R}}^n . f(\alpha_1,..,\alpha_n) \in PT_{\mathcal{R}}$$

and arrows [combining the operations on $T_{\Sigma/E}$ and $PT[\mathcal{R}]$:

$$f_{Spec_{\mathcal{R}}} =_{def} \lambda([u_1] \xrightarrow{\alpha_1} [v_1], \ldots, [u_n] \xrightarrow{\alpha_n} [v_n]) . [f(u_1,..,u_n)] \xrightarrow{f(\alpha_1,..,\alpha_n)} [f(v_1,..,v_n)] \in Spec_{\mathcal{R}}$$

and it has also idle arrows $[t] \xrightarrow{t} [t]$ for $t \in T_{\Sigma}$, and a sequential arrow composition operation:

$$-;- : ([u] \xrightarrow{\alpha} [v], [v] \xrightarrow{\beta} [w]) \mapsto [u] \xrightarrow{\alpha;\beta} [w]$$

Step 2: Specifying the concurrent computations of the system defined by $\mathcal{R}$ as the arrows of the category $\mathcal{T}_{\mathcal{R}}$

This second step gives a full answer to the ten-million-dollar question:

> How can we naturally model the concurrent computations of a general concurrent system?

and in particular to the question:

> when are two ~~concurrent~~ specifications of concurrent computations specifications of the same concurrent computation?

This accomplished by defining a quotient labeled graph

$\mathcal{T}_{\mathcal{R}} = (T_{\Sigma/E}, PT_{R/\equiv}, \overrightarrow{\mathcal{T}_{\mathcal{R}}})$ of $Spec_{\mathcal{R}}$, where the congruence relation $\equiv$ (respecting the Σ-operations, and $-;-$ in $Spec_{\mathcal{R}}$) identifying both labels and arrows is the smallest congruence generated by the following equivalences for $\mathcal{R} = (\Sigma, E, R)$:

1. E-Equality. For each $(u(x_1,..,x_n) = v(x_1,..,x_n)) \in E$ and $[w_1] \xrightarrow{\alpha_1} [w_1'], \ldots, [w_n] \xrightarrow{\alpha_n} [w_n']$ in $\mathrm{Spec}\mathcal{R}$

- $u(\alpha_1,..,\alpha_n) \equiv v(\alpha_1,..,\alpha_n)$ and therefore (because $T_{\Sigma/E} \models E$)
- $\left([u(w_1,..,w_n)] \xrightarrow{u(\alpha_1,..,\alpha_n)} [u(w_1',..,w_n')]\right) \equiv \left([v(w_1,..,w_n)] \xrightarrow{v(\alpha_1,..,\alpha_n)} [v(w_1',..,w_n')]\right)$

2. Category. For each $[u_1] \xrightarrow{\alpha} [u_2], [u_2] \xrightarrow{\beta} [u_3], [u_3] \xrightarrow{\gamma} [u_4]$ in $\mathrm{Spec}\mathcal{R}$,

- $\left([u_1] \xrightarrow{(\alpha;\beta);\gamma} [u_4]\right) \equiv \left([u_1] \xrightarrow{\alpha;(\beta;\gamma)} [u_4]\right)$
- $\left([u_1] \xrightarrow{u_1;\alpha} [u_2]\right) \equiv \left([u_1] \xrightarrow{\alpha} [u_2]\right) \equiv \left([u_1] \xrightarrow{\alpha;u_2} [u_2]\right)$

which, of course, follows from the proof-term equivalences: $(\alpha;\beta);\gamma \equiv \alpha;(\beta;\gamma)$, and $u_1;\alpha \equiv \alpha \equiv \alpha;u_2$ for the above labeled arrows.

3. Functoriality. For each $n \geq 1$, $f \in \Sigma_n$, and

$$\left.\begin{array}{l} [u_1] \xrightarrow{\alpha_1} [v_1], \ldots, [u_n] \xrightarrow{\alpha_n} [v_n] \\ [v_1] \xrightarrow{\beta_1} [w_1], \ldots, [v_n] \xrightarrow{\beta_n} [w_n] \end{array}\right\} \text{in } \mathrm{Spec}\mathcal{R}$$

$$\left([f(u_1,..,u_n)] \xrightarrow{f(\alpha_1;\beta_1,..,\alpha_n;\beta_n)} [f(w_1,..,w_n)]\right) \equiv \left([f(u_1,..,u_n)] \xrightarrow{f(\alpha_1,..,\alpha_n);f(\beta_1,..,\beta_n)} [f(w_1,..,w_n)]\right)$$

Exchange. For each $l : t(x_1,..,x_n) \to t'(x_1,..,x_m)$ in R and

$[u_1] \xrightarrow{\alpha_1} [v_1], \dots, [u_m] \xrightarrow{\alpha_m} [v_m]$ in Spec_R

$$\left([t(u_1,..,u_n)] \xrightarrow{l(u_1,..,u_n);\, t'(\alpha_1,..,\alpha_m)} [t'(v_1,..,v_m)]\right) \equiv \left([t(u_1,..,u_n)] \xrightarrow{l(\alpha_1,..,\alpha_m)} [t'(v_1,..,v_n)]\right) \equiv$$

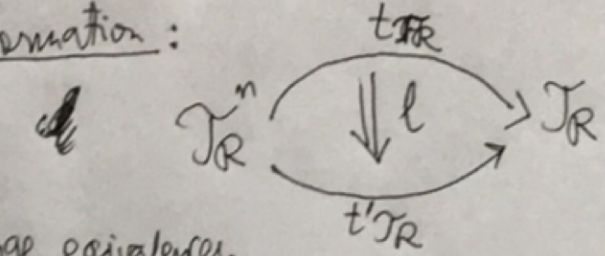
$$\equiv \left([t(u_1,..,u_n)] \xrightarrow{t(\alpha_1,..,\alpha_n);\, l(v_1,..,v_n)} [t'(v_1,..,v_m)]\right)$$

Main Theorem $\mathcal{T}_R$ is a category. Furthermore, it has a Σ-algebra structure as a category which satisfies the equations E. This is obvious for nodes and labels because of E-equality, and holds also for arrows by the operation

$$f_{\mathcal{T}_R} =_{def} \lambda([u_1] \xrightarrow{[\alpha_1]} [v_1], \dots, [u_n] \xrightarrow{[\alpha_n]} [v_n]) \in \mathcal{T}_R^n \cdot [f(u_1,..,u_n)] \xrightarrow{[f(\alpha_1,..,\alpha_n)]} [f(v_1,..,v_m)] \in \mathcal{T}_R$$

which is a functor $f_{\mathcal{T}_R} : \mathcal{T}_R^n \to \mathcal{T}_R$ because of the Functoriality equivalences, i.e., $\mathcal{T}_R \models E$ as a category.

This furthermore means that for each $l : t(x_1,..,x_n) \to t'(x_1,..,x_m)$ in R we have functors $t_{\mathcal{T}_R}, t'_{\mathcal{T}_R} : \mathcal{T}_R^n \to \mathcal{T}_R$ and a natural transformation:



because of the Exchange equivalences.