Defining P systems in K

Traian Florin Șerbănuță[†], Gheorghe Ștefănescu[‡], and Grigore Roșu[†]

[†]University of Illinois at Urbana-Champaign
[‡]University of Bucharest

July 31, 2008

Introduction

- First formal definition of P systems in a specification language
  - Definition, not implementation or encoding
  - One-to-one correspondence of rules
  - Almost zero representation distance
- Embedding into K gives possibilities for extensions of P systems
  - Objects with algebraic structure, satisfying global axioms and rules
  - Membranes with nucleus – modeling DNA transcription
- K comes with the plethora of formal tools from Maude
  - Rewriting engine, state space exploration
  - LTL model checking, theorem prover

Outline

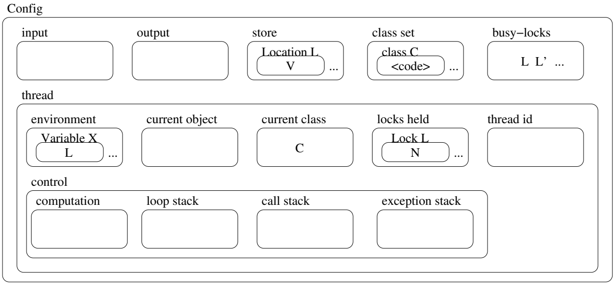
K: a rewriting-based framework for computations

- Based on list and multiset term rewriting
- Inherits Rewriting Logic concepts of equations and rules
  - Configurations are equivalence classes of terms
  - Rules are used to transit between states
- Specialized notation to improve readability and modularity
- Parallelism with sharing of data (e.g., concurrent reads)
- Was used to define highly non-trivial programming languages
  - Java 1.4, Scheme, Haskell
- Q: Can K model a highly dynamic execution environment?

K configurations: nested multisets

- Configuration items are encapsulated into labeled “cells”
  - Notation: $\llparenthesis \texttt{<contents>} \rrparenthesis_{\texttt{<label>}}$
  - One to one correspondence to membrane structures

Example: Configuration of KOOL, an OO language



K optimized notation for rules

K Contexts avoid repeating information not changed by the rule

- Assignment rule using multiset&list rewriting:

$$(\!| x := v \curvearrowright k |\!)_k \; (\!| (\!| v' |\!)_x \; s |\!)_{state} \longrightarrow (\!| k |\!)_k \; (\!| (\!| v |\!)_x \; s |\!)_{state}$$

- Assignment using K contexts: $(\!| \underbrace{x := v}_{\cdot} \curvearrowright k |\!)_k \; (\!| (\!| \underbrace{v'}_{v} |\!)_x \; s |\!)_{state}$

Angle brackets simplify list and set matching

- Use $(\!| \, \rangle$, $\langle \, |\!)$, and $\langle \, \rangle$ to match prefixes, suffixes, and subsets

$$(\!| \underbrace{x := v}_{\cdot} \rangle_k \; \langle (\!| \underbrace{\_}_{v} |\!)_x \rangle_{state}$$

- Anonymous variables replace unchanged or not-needed variables

Good notation yields good results – Parallelism in K

- SOS definitions enforce interleaving semantics
- Rewriting logic (CHAM similarly) more parallel, but still not enough
  - Concurrent accesses to the state are disallowed

    e.g., $(\!| x := v \curvearrowright k |\!)_k \; (\!| (\!| v' |\!)_x \; s |\!)_{state} \;\longrightarrow\; (\!| k |\!)_k \; (\!| (\!| v |\!)_x \; s |\!)_{state}$
- K allows concurrent modifications with shared structure
  - As long as they do not change their shared part

    $$\langle\!| \underbrace{x := v}_{\cdot} |\!\rangle_k \;\; \langle\!| (\!| \underbrace{\_}_{v} |\!)_x \; |\!\rangle_{state}$$
  - Extremely usefull when defining promoters & inhibitors

Outline

Basic transition P systems

- A membrane $[_h]_h$ with contents x is represented as $\langle\!|x|\!\rangle_h$
  - Here concatenation is a multiset constructor (ACU)
  - x can contain other membranes, as well as common objects
- A rule in $[_h]_h$: $u \longrightarrow v_{here} v_{out} \prod_{i=1}^{k} v_{in_{h_i}}$ becomes a global rule

$$\langle\!| \frac{u}{v_{here}} \quad \langle\!| \frac{\cdot}{v_{in_{h_1}}} |\!\rangle_{h_1} \cdots \langle\!| \frac{\cdot}{v_{in_{h_k}}} |\!\rangle_{h_k} |\!\rangle_h \quad \frac{\cdot}{v_{out}}$$

- δ is considered to have *here* attribute
  - Configuration is globally normalized by equations

$$\delta\delta \rightharpoonup \delta \text{ and } (\!|x\delta|\!)_h \rightharpoonup x$$

Catalysts vs. Promoters/Inhibitors

- Catalysts. *c* is required for *a* to become *b*: $\langle\!\underline{ac}\!\rangle_h$ over bc, i.e. $\left\langle \frac{ac}{bc} \right\rangle_h$
- Promoters. *a* becomes *b* if *c* is present: $\left\langle \frac{a}{b}\ c \right\rangle_h$
  - In RWL, the two rules above would be identical
  - P systems add special syntax to distinguish promoters
  - In K, *c* actively participates to first rule, passively to the second
- Inhibitors. *a* becomes *b* if *c* is not present: $\left\langle \frac{a}{b}\ x \right\rangle_h \ \texttt{ if } \ c \notin x$

Other variations of transition P systems

Polarities

- Pair each data/membrane label with its polarity. E.g.,
  a polarity changing rule: $\langle u \rangle_h^+ \to \langle u \rangle_h^-$ or, equivalently, $\langle u \rangle_h^{\frac{+}{-}}$

Membrane Permeability allows for impenetrable membranes

- Pair membrane labels with permeability indexes
- Check permeability through matching: $\langle u \rangle_h^1 \to \ldots$
- Disolve membranes with permeability 0: $(\!| x |\!)^0_{\_} \rightharpoonup x$

Basic symport/antiport P systems

- Same assumptions as in the original setting: one membrane, initial configuration $\langle\!| l_0 |\!\rangle$, potentially infinite environment
- Antiport rule $((u, out), (v, in))$ becomes in K $\langle\!| \frac{u}{v} |\!\rangle \ \frac{v}{u}$
  - $l_1 : (\texttt{inc}(r), l_2, l_3)$ is represented as $\langle\!| \frac{l_1}{a_r l_2} |\!\rangle \ \frac{a_r l_2}{l_1}$ and $\langle\!| \frac{l_1}{a_r l_3} |\!\rangle \ \frac{a_r l_3}{l_1}$
- Symport rules (u, out) and (v, in) become in K $\langle\!| \frac{u}{\cdot} |\!\rangle \ \frac{\cdot}{u}$ and $\langle\!| \frac{\cdot}{u} |\!\rangle \ \frac{u}{\cdot}$
  - $l_1 : \texttt{halt}$ is represented as $\langle\!| \frac{l_1}{\cdot} |\!\rangle \ \frac{\cdot}{l_1}$

P systems with active membranes

- Object evolution rule $[_h a \rightarrow v]_h^e$ becomes in K: $\langle \underline{a} \rangle_h^e$ over v
- in communication rule $a[_h]_h^{e_1} \rightarrow [_h b]_h^{e_2}$ becomes in K: $\frac{a}{\cdot}$ $\langle \frac{\cdot}{b} \rangle_h^{\frac{e_1}{e_2}}$
- out communication rule $[_h a]_h^{e_1} \rightarrow [_h]_h^{e_2} b$ becomes in K: $\langle \frac{a}{\cdot} \rangle_h^{\frac{e_1}{e_2}}$ $\frac{\cdot}{b}$
- dissolving rule $[_h a]_h^e \rightarrow b$ becomes in K: $(\!| a \quad x |\!)_h^e \rightarrow b \quad x$
- division rule for elementary membranes,
  $[_h a]_h^{e_1} \rightarrow [_h b]_h^{e_2} [_h c]_h^{e_3}$ becomes in K: $(\!| a |\!)_h^{e_1} \rightarrow (\!| b |\!)_h^{e_2} \quad (\!| c |\!)_h^{e_3}$

Active membranes variations

- **Membrane creation.**

$$a \longrightarrow (\!| v |\!)_h$$

- **Merging of membranes.**

$$(\!| x |\!)_h \quad (\!| y |\!)_{h'} \longrightarrow (\!| z |\!)_{h''}$$

- **Split of membranes.**

$$(\!| z |\!)_{h''} \longrightarrow (\!| x |\!)_h \quad (\!| y |\!)_{h'}$$

- **Endocytosis and exocytosis.**

$$\frac{(\!| x |\!)_h}{\cdot} \quad \left(\!\left| \frac{\cdot}{(\!| y |\!)_h} \right|\!\right)_{h'}$$

Outline

Improvements of K suggested by P systems

Priorities and other strategies

- Capturing control mechanisms is a matter of strategies
- K does not currently employ strategies

Arbitrary Jumps directly move an object into another membrane. In K:

- if membranes don't contain each other, $\langle\!\langle\!\langle^{*}\ \frac{u}{\cdot}\ {}^{*}\rangle\!\rangle_{h}\quad \langle\!\langle^{*}\ \frac{\cdot}{v}\ {}^{*}\rangle\!\rangle_{h'}\rangle\!\rangle$
- if the jump is into an enclosed membrane, then $\langle\!\langle \frac{u}{\cdot}\quad \langle\!\langle^{*}\ \frac{\cdot}{v}\ {}^{*}\rangle\!\rangle_{h'}\rangle\!\rangle_{h}$

Gemmation Encapsulate into $[_{@h}\ ]_{@h}$ a to be carried to $[_{h}\ ]_{h}$.

- $[_{@h}u]_{@h}$ travels through the system, one membrane at a time.
- Maintaining dynamic structure information can solve this problem

Conclusions

- K seems powerful enough to capture the parallelism of P systems
- Suitable as a definition/implementation medium for P systems
  - Granularity of computation can be preserved
  - Almost zero-representation distance
  - Use tools available for K through Maude
- Opens new lines of research for P systems
  - P systems with structured objects
  - Dynamic rule generation – modeling DNA transcription?