A Truly Concurrent Semantics for the $\mathbb{K}$ Framework Based on Graph Transformations

Traian Florin Șerbănuță and Grigore Roșu

University “Alexandru Ioan Cuza” Iași
University of Illinois at Urbana-Champaign

The 𝕂 Framework

http://k-framework.org

What is K?

A tool-supported rewrite-based framework for defining programming language design and semantics.

Why?

- Programming languages must have formal semantics!
- And analysis/verification tools should build on them
  - Otherwise they are adhoc and likely wrong

The $\mathbb{K}$ Framework

Defining programming languages

- Java 1.4 (Chen, CAV'06)
- Scheme (Hills&Meredith, SCHEME'07)
- Verilog (Meredith&Katelman, MEMOCODE'10)
- C (Chucky Ellison, POPL'12)
- In progress: Haskell, LLVM, Javascript, ...
- Paradigmatic teaching languages (functional, object-oriented, agents-based)

The 𝕂 Framework

Tool support

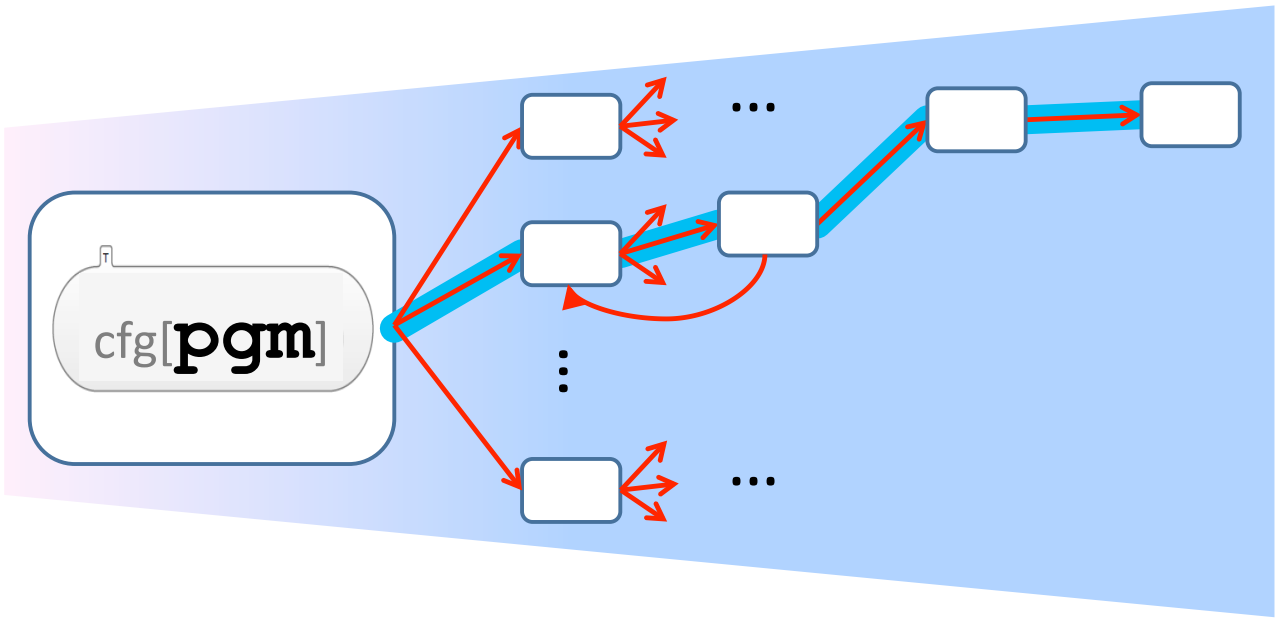
- Efficient and interactive execution (interpreters)
- State-space exploration (search and model-checking)
- Deductive program verification (in progress)

Leveraging the generic tool support given by the Maude rewrite engine

The 𝕂 Framework

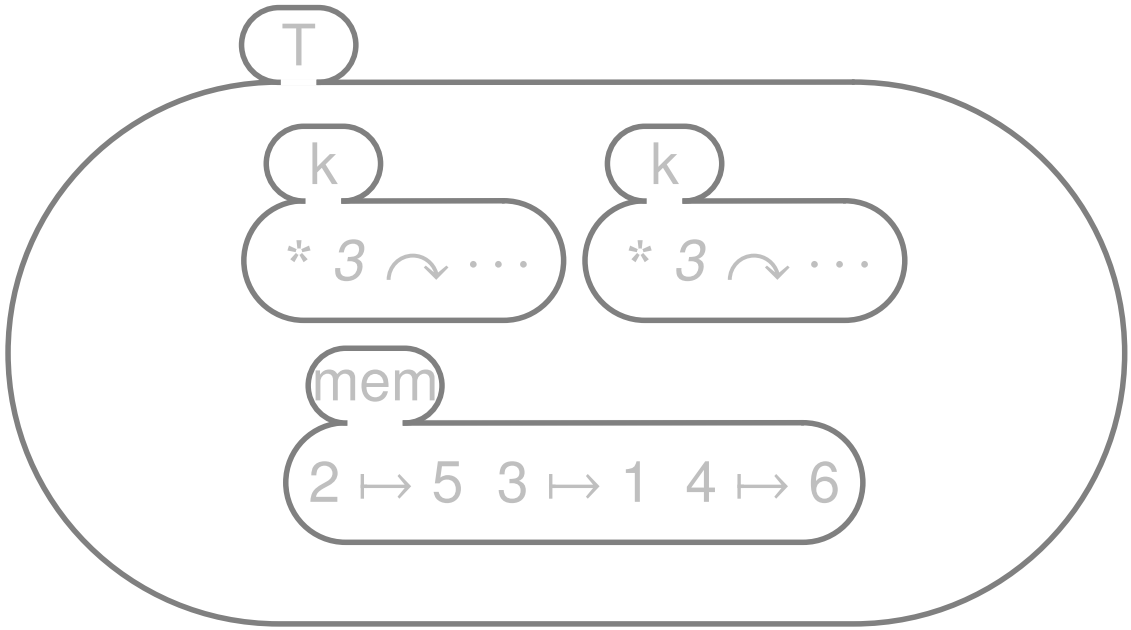
Rewriting based

- Running configurations represented as first order terms
- Rules specify allowed transitions between configurations
- Semantics as a transition system



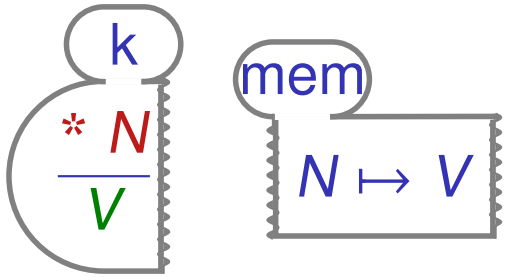
$\mathbb{K}$ configurations and rules

Running configuration

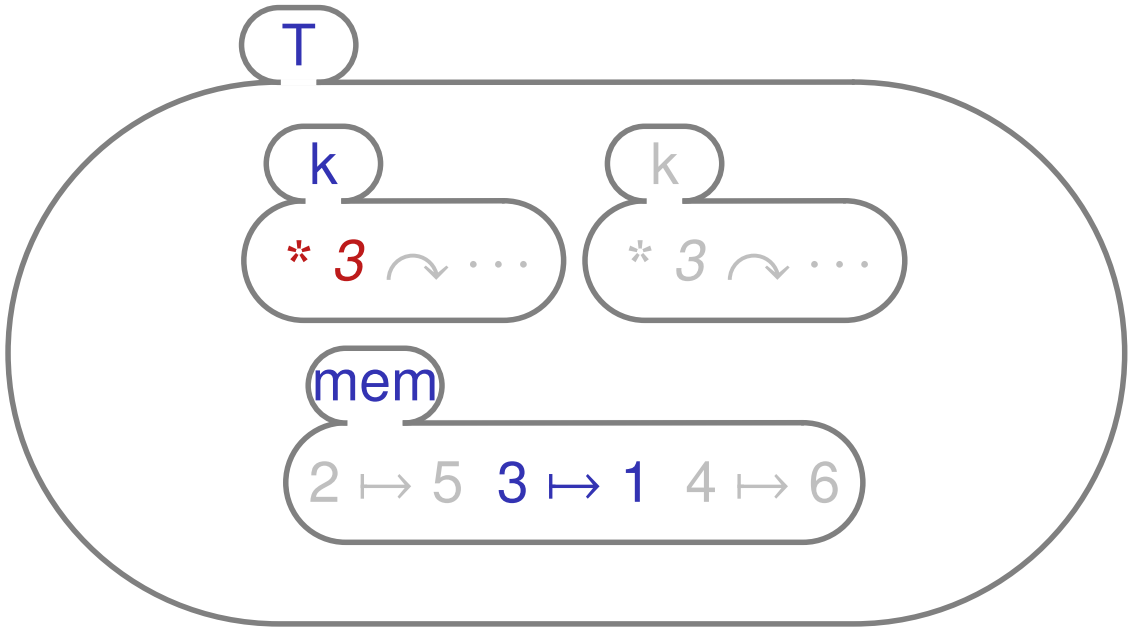


$\mathbb{K}$ configurations and rules

Read rule



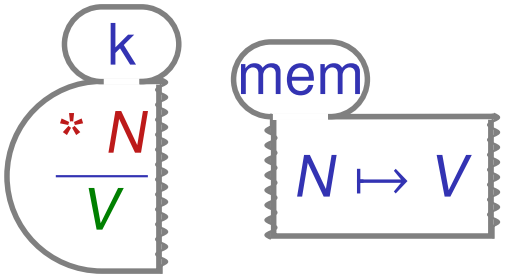
Running configuration



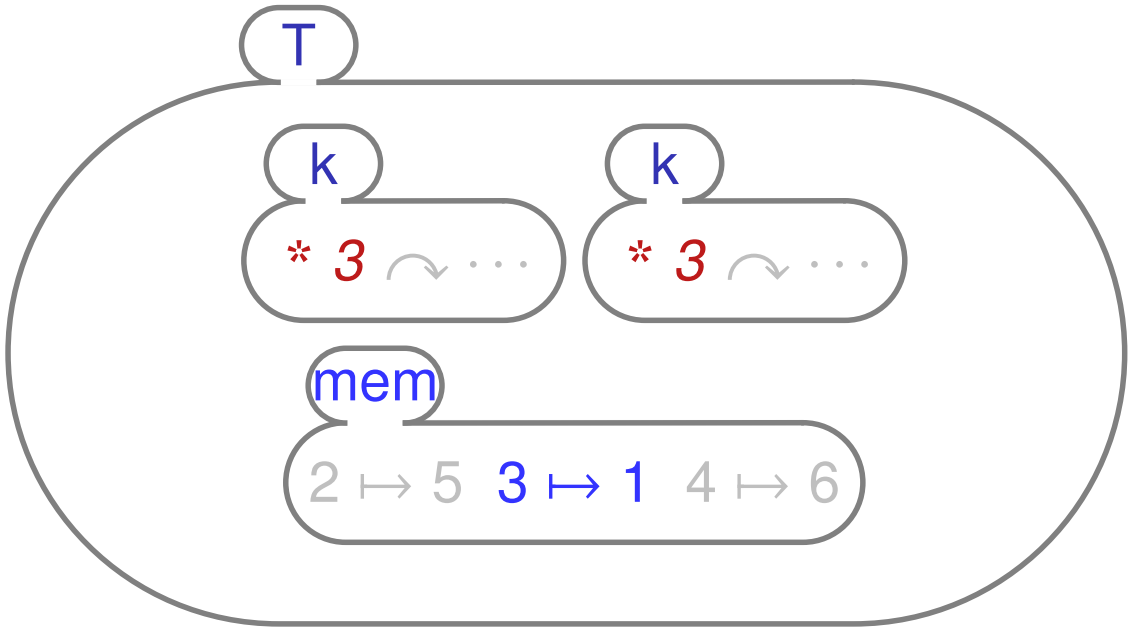
$\mathbb{K}$ configurations and rules

More concurrency with $\mathbb{K}$ rules?

Read rule



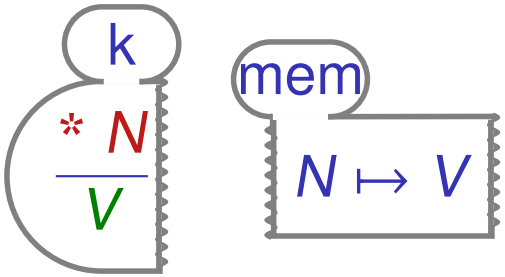
Running configuration



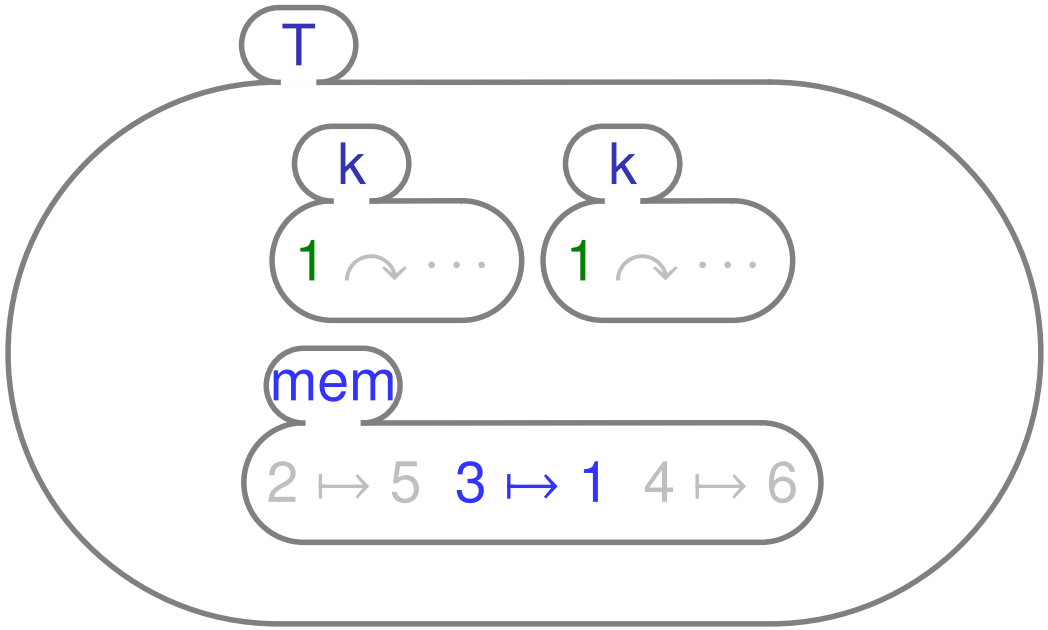
$\mathbb{K}$ configurations and rules

More concurrency with $\mathbb{K}$ rules!

Read rule



Running configuration



Semantics Requirements

- Conservative extension of term rewriting
- While allowing as much concurrency as possible

(1) $h(\frac{x}{g(x,x)}, y, \underline{1})$ (2) $h(x, \underline{0}, y)$ (3) $\underline{a}$ (4) $\underline{f(x)}$

$$h(f(a), 0, 1) \xRightarrow{(1)+(2)+(3)+(4)} h(g(b, b), 1, 0)$$

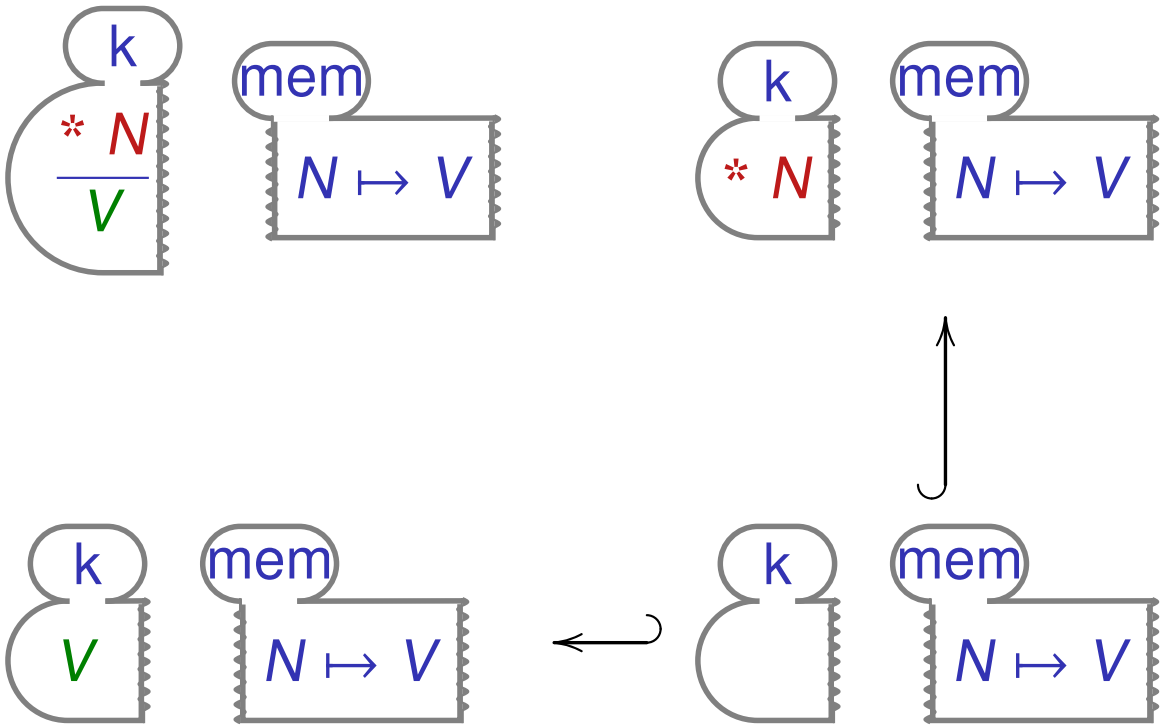
$\mathbb{K}$ rules resemble graph transformation rules

The DPO approach

Graph transformation rule (DPO)

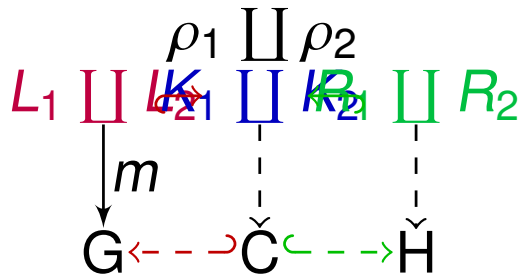
$$K \cup L \hookleftarrow K \hookrightarrow K \cup R$$

$\mathbb{K}$ rule



Concurrency and serializability in graph transformations

$\rho_1 : L_1 \hookleftarrow K_1 \hookrightarrow R_1$ $\rho_2 : L_2 \hookleftarrow K_2 \hookrightarrow R_2$



Theorem (Parallelism and serializability [*Ehrig, Kreowski, 1976*]**)**

If $G \overset{m,\rho_1}{\Longrightarrow} H_1$, *and* $G \overset{m,\rho_2}{\Longrightarrow} H_2$ *are parallel independent,i.e., only overlapping on the read-only part, then (1)* $G \overset{m,\rho_1 \amalg \rho_2}{\Longrightarrow\!\!\!\Longrightarrow} H$ *(concurrency); and (2)* $H_1 \overset{\rho_2}{\Longrightarrow} H$ *and* $H_2 \overset{\rho_1}{\Longrightarrow} H$ *(serializability).*

Formally capturing the concurrency of $\mathbb{K}$

Given that . . .

- $\mathbb{K}$ rules resemble graph transformation rules
- Graph rewriting captures concurrency with sharing of context

Capture the concurrency intended for $\mathbb{K}$ through graph rewriting

- Term graph rewriting approaches seem a promising start
  - Are sound and complete w.r.t. term rewriting
  - Are special forms of graph transformations
- However, term graph rewriting have 0-sharing (like term rewriting)

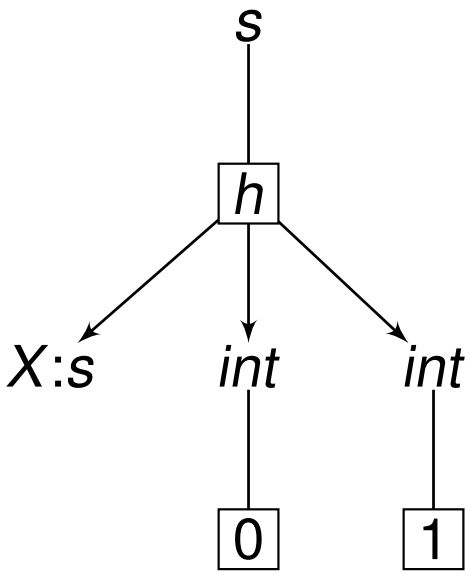
Representing terms as graphs

Term

$$h(X, 0, 1)$$

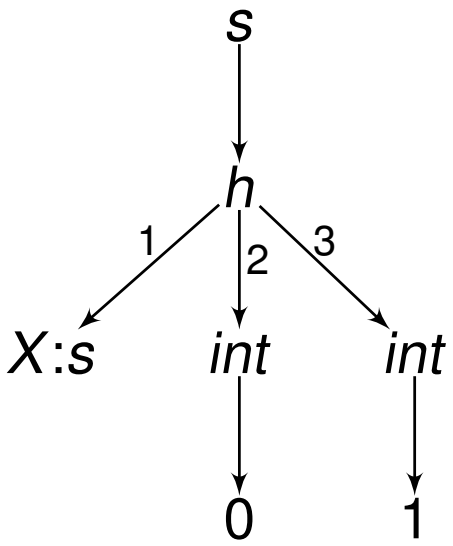
where X is a variable, h and X are of sort s, and 0 and 1 are integers

Jungle (hypergraph) representation



[*Habel, Kreowski, Plump, 1987*]

Graph representation



$\mathbb{K}$ graph rules: a new kind of term graph rewriting rules

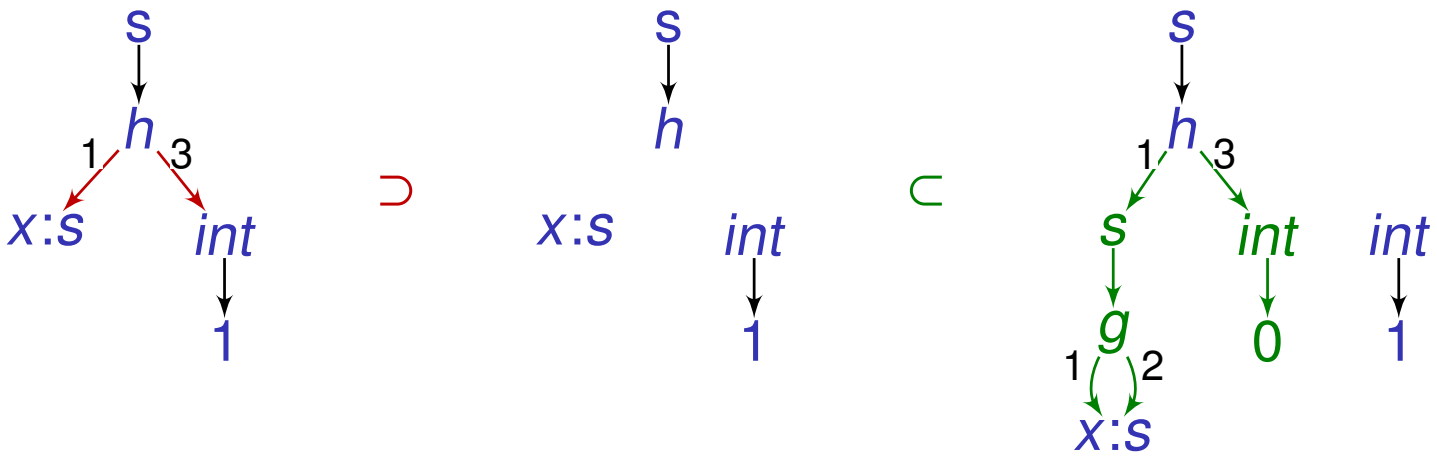
$\mathbb{K}$ rule ρ

$$h(\underset{g(x,x)}{x}, y, \underset{0}{1})$$

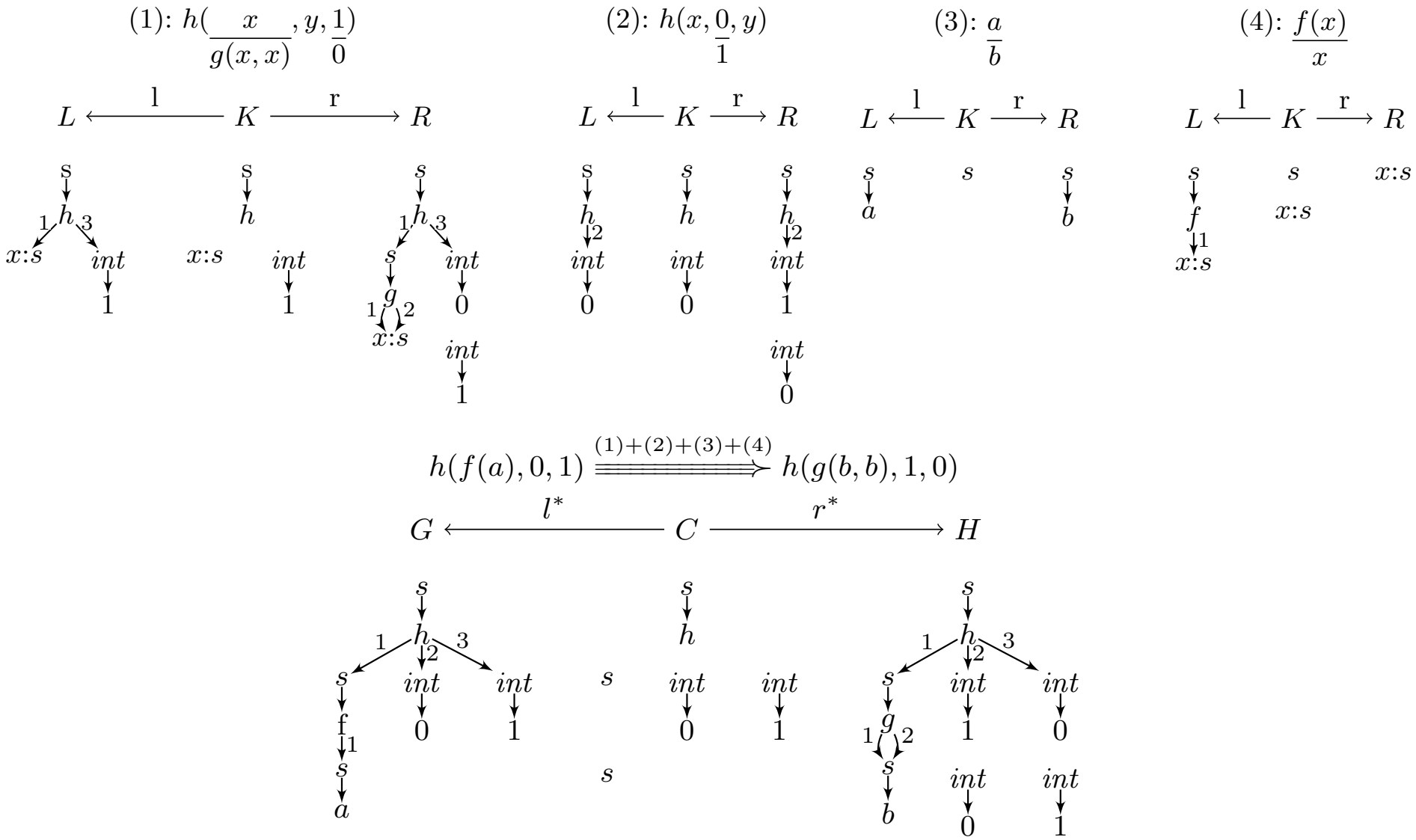
Direct representation as a rewrite rule $K2R(\rho)$

$$h(x, y, 1) \rightarrow h(g(x, x), y, 0)$$

Corresponding graph rewrite rule $K2G(\rho)$



Desired level of parallelism



$\mathbb{K}$ rewriting

Definition

Let $\mathbb{S}$ be a $\mathbb{K}$ rewrite system and t be a term. Then

$$t \underset{\mathbb{K}}{\overset{\mathbb{S}}{\Longrightarrow}} t' \quad \text{iff} \quad K2G(t) \xrightarrow[Graph]{K2G(\mathbb{S})} H \text{ such that } \mathrm{term}(H) = t'$$

Theorem (Correctness w.r.t. rewriting)

Soundness: *If* $t \underset{\mathbb{K}}{\overset{\rho}{\Longrightarrow}} t'$ *then* $t \xrightarrow[Rew]{K2R(\rho)} t'$.

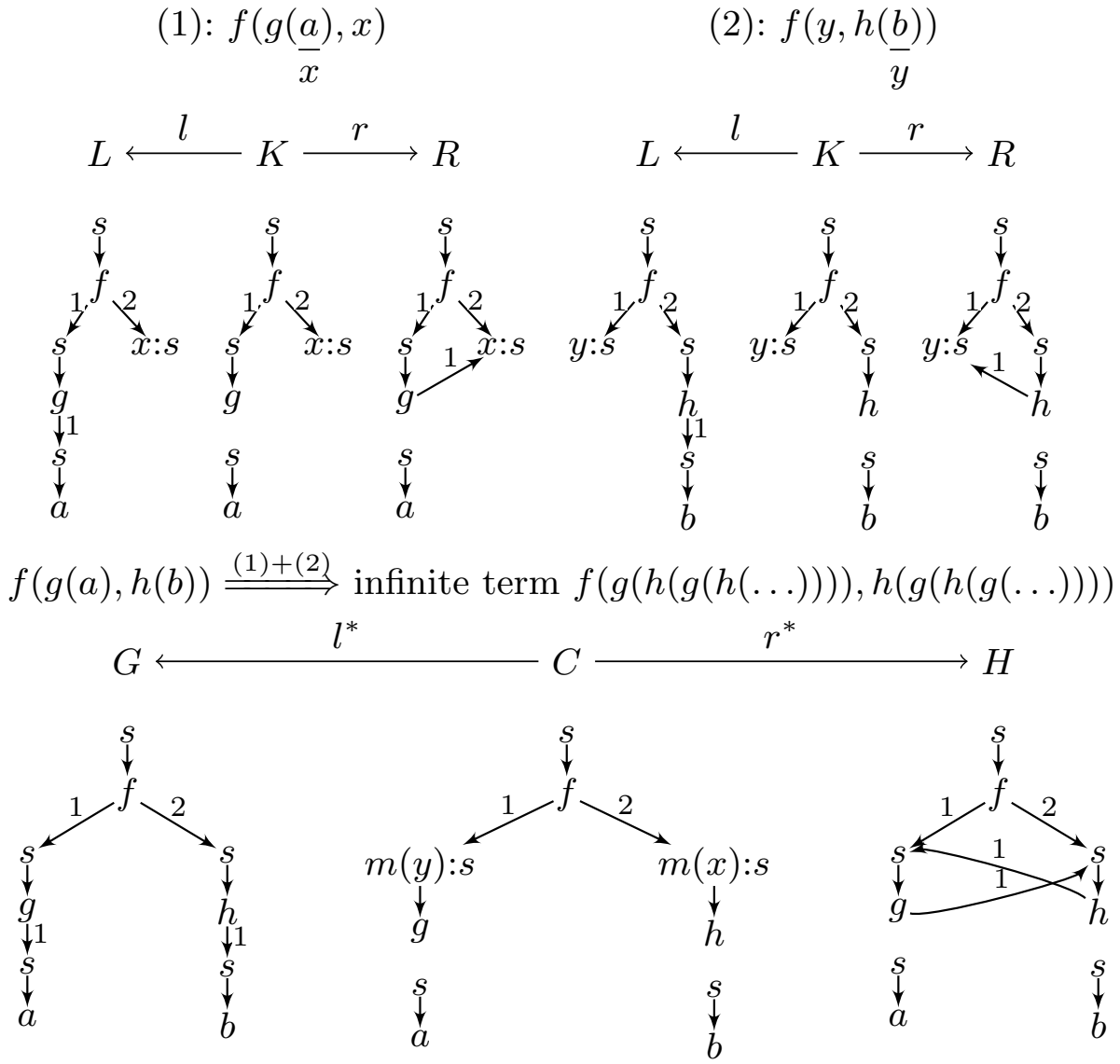
Completeness: *If* $t \xrightarrow[Rew]{K2R(\rho)} t'$ *then* $t \underset{\mathbb{K}}{\overset{\rho}{\Longrightarrow}} t'$.

Serializability: *If* $t \underset{\mathbb{K}}{\overset{\rho_1+\cdots+\rho_n}{\Longrightarrow}} t'$ *then* $t \underset{\mathbb{K}}{\overset{\rho_1{}^*}{\Longrightarrow}} \cdots \underset{\mathbb{K}}{\overset{\rho_n{}^*}{\Longrightarrow}} t'$ *(and thus,* $t \xrightarrow[Rew]{*} t'$*).*

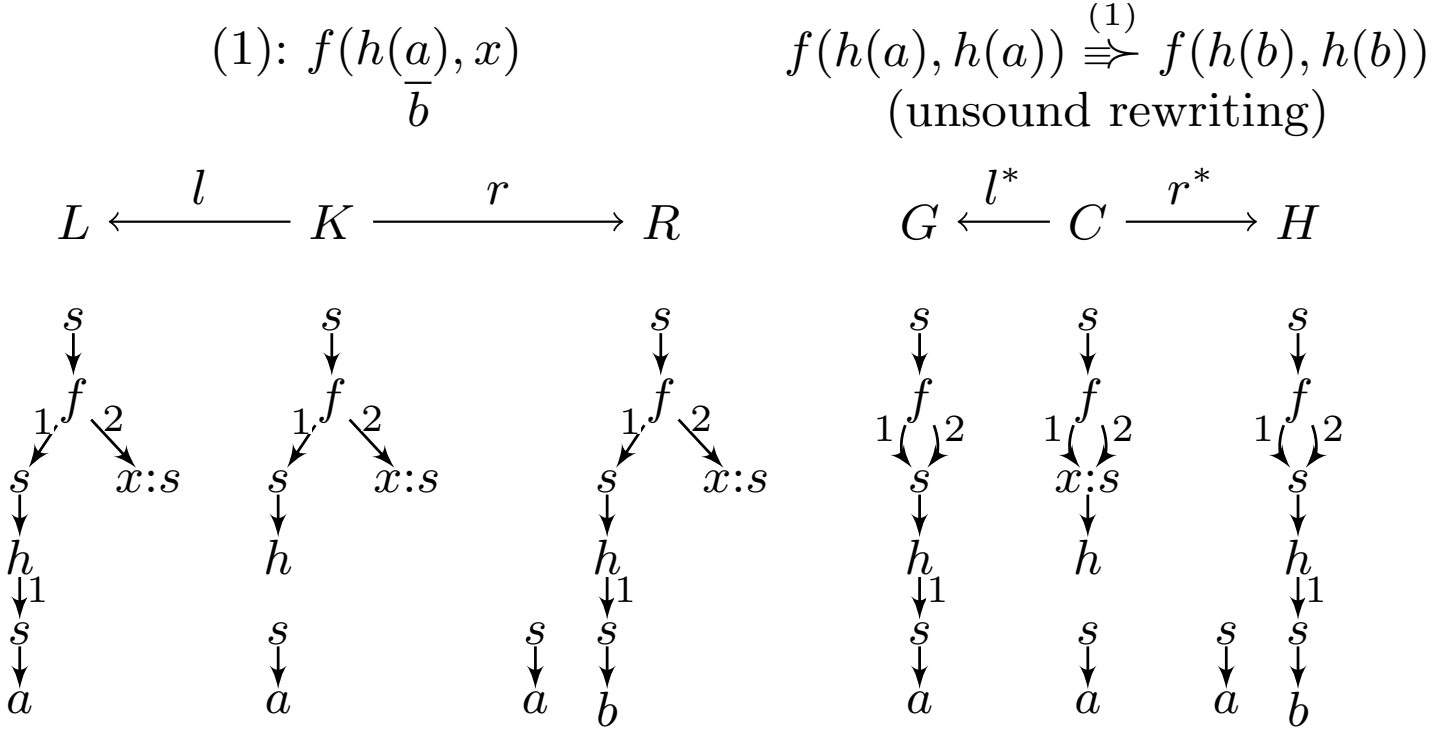
Instead of proof

- $\mathbb{K}$ term graphs are stable under concurrent applications of $\mathbb{K}$ rules
  - If their instances only overlap on read-only part
  - If they do not introduce cycles
- Serializability based on graph rewriting serializability
- $\mathbb{K}$ graph rewriting conservatively extends Jungle rewriting
  - For terms without subterm sharing
- Jungle rewriting is sound and complete w.r.t. rewriting
  [*Holland, Plump, 1991; Plump 1999*]

Parallel $\mathbb{K}$ graph rewriting can introduce cycles



Subterm sharing can jeopardize soundness



Conclusions

Results: A new formalism of term-graph rewriting

- Sound and complete w.r.t. term rewriting
- Capturing the intended concurrency of $\mathbb{K}$ rewriting

Future work

- Investigate the cycle condition
- Special graph representations for lists and multisets
  - And re-prove the correctness for this representation
- Tools which take advantage of this new semantics?