Universität Rostock Traditio et Innovatio

visual & analytic computing

Nonlinear pattern matching in rule-based modeling languages

Tom Warnke and Adelinde M. Uhrmacher
Institute for Visual and Analytic Computing
University of Rostock

Rule-based modeling languages

Stochastic graph rewriting: Kappa, BNGL

Stochastic term rewriting: (C)SMMR, ML-Rules, React(C), Chromar

$$A(x) + A(y) \xrightarrow{k} A(x + y)$$

Rule-based modeling languages

Stochastic graph rewriting: Kappa, BNGL

Stochastic term rewriting: (C)SMMR, ML-Rules, React(C), Chromar

reactants — rate — products

$$\overbrace{A(x) + A(y)}^{\text{reactants}} \xrightarrow{k} \overbrace{A(x+y)}^{\text{products}}$$

species name — variable for attribute

Rule-based modeling languages

Stochastic graph rewriting: Kappa, BNGL

Stochastic term rewriting: (C)SMMR, ML-Rules, React(C), Chromar

reactants — rate — products

$$\overbrace{A(x) + A(y)}^{\text{reactants}} \xrightarrow{k} \overbrace{A(x + y)}^{\text{products}}$$

A: species name; x: variable for attribute; k: rate

Match in solution

$\{\!|A(1), A(2), \ldots|\!\}$

Rule-based modeling languages

Stochastic graph rewriting: Kappa, BNGL

Stochastic term rewriting: (C)SMMR, ML-Rules, React(C), Chromar

reactants — rate — products

$$\overbrace{A(x) + A(y)}^{\text{reactants}} \xrightarrow{k} \overbrace{A(x+y)}^{\text{products}}$$

species name — variable for attribute

Match in solution $\{\!|A(1), A(2), \ldots|\!\}$

$$A(1) + A(2) \xrightarrow{k} A(3)$$

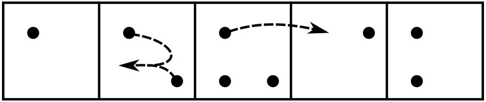
$$A(1) + A(3) \xrightarrow{k} A(4)$$

$$\ldots$$

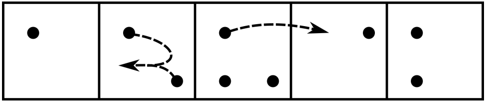
Overview

- Rules use patterns on their left side, which are matched to the current solution to obtain reactions.
- Nonlinear patterns are patterns in which variables may occur multiple times.
- In this talk:
    - Nonlinear patterns are useful for expressing relations between reactants (e.g., spatial relations).
    - State-of-the-art languages only support linear patterns.
    - Supporting nonlinear patterns allows more efficient pattern matching.

The case for nonlinear patterns

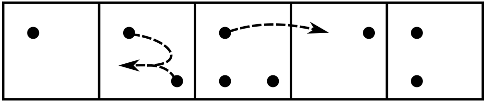


The case for nonlinear patterns



$$A(x) + A(x) \to \emptyset$$
$$A(x) \to A(x+1)$$

The case for nonlinear patterns



$$A(x) + A(x) \rightarrow \emptyset$$
$$A(x) \rightarrow A(x+1)$$

$$C(x)[A + A + r] \rightarrow C(x)[r]$$
$$C(x)[A + r_1] + C(x+1)[r_2] \rightarrow C(x)[r_1] + C(x+1)[A + r_2]$$

State of the art

Expressing patterns

State-of-the-art languages only allow linear patterns, as linear pattern matching is simpler than nonlinear pattern matching.

In these languages, nonlinear patterns must be linearized to express them, resulting in rules with constraints.

$$A(x) + A(x) \longrightarrow \emptyset$$

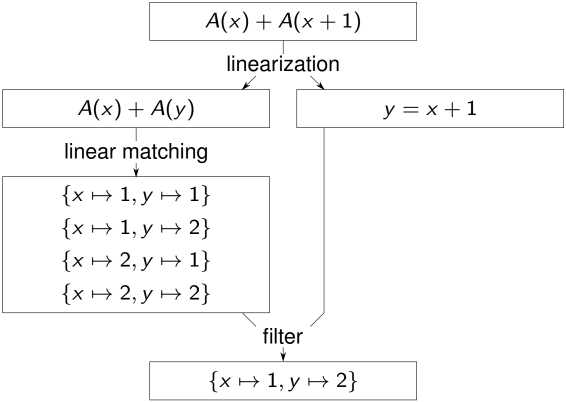
$$A(x) + A(y) \xrightarrow[x=y]{} \emptyset$$

$$C(x)[A + r_1] + C(x + 1)[r_2] \longrightarrow C(x)[r_1] + C(x + 1)[A + r_2]$$

$$C(x)[A + r_1] + C(y)[r_2] \xrightarrow[y=x+1]{} C(x)[r_1] + C(y)[A + r_2]$$

State of the art

Matching $A(x) + A(x+1)$ in the solution $\{\!\!\{A(1), \ldots, A(n)\}\!\!\}$ with $n = 2$



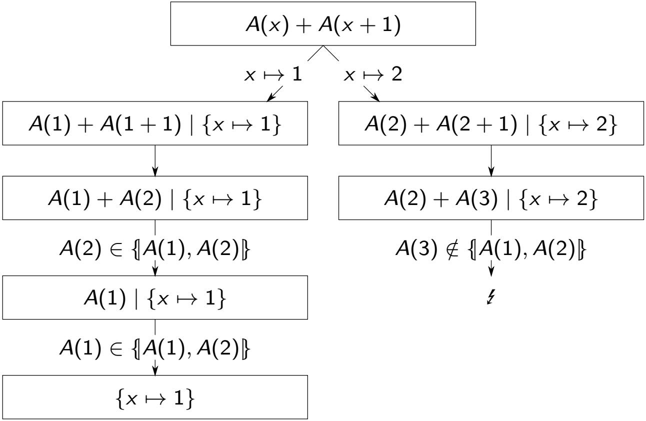
State of the art

Problem

- In a solution $\{\!|A(1), \ldots, A(n)|\!\}$ we get $n - 1$ eventual matches, but n^2 intermediate results.
- Most intermediate results are thrown away.
- The relation between different occurrences of the same variable is not available during pattern matching.

Inline substitution

Matching $A(x) + A(x+1)$ in the solution $\{\!|A(1), \ldots, A(n)|\!\}$ with $n = 2$



Inline substitution

Idea

Try all possible values for a variable

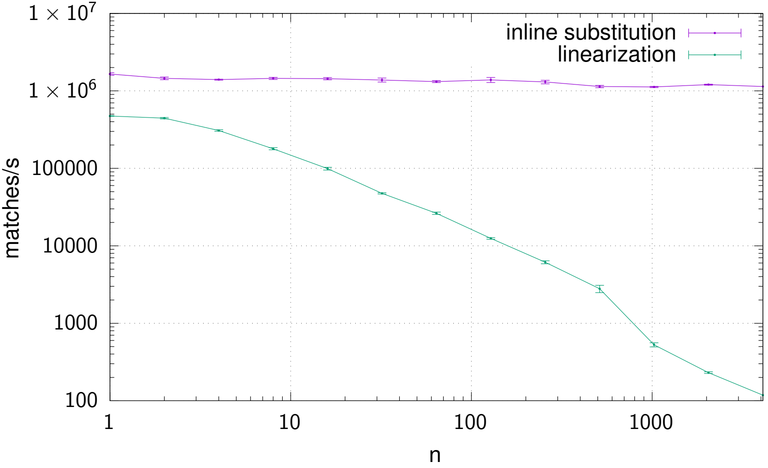
- The chosen value is substituted for all occurrences of the variable.

$\Rightarrow$ We can evaluate expressions.

$\Rightarrow$ We can quickly check whether the reactants exist.

$\Rightarrow$ We can prune unsuccessful branches early.

Matching $A(x) + A(x+1)$ in the solution $\{\!|A(1), \ldots, A(n)|\!\}$



Context

- Nonlinear patterns are unpopular in term rewriting and functional programming.
- Logic programming languages (e.g., Prolog) allow nonlinear patterns, but have limited support for expressions.
- Functional logic programming languages (e.g., Curry) are a current research topic.
- We need to count match multiplicities for mass action kinetics.

Conclusion and open questions

- Usually, a more expressive language means less efficient execution.
- However, nonlinear patterns contain valuable information about the modeler intent, enabling faster algorithms.
- Can we relate term rewriting and graph rewriting?
- Is pattern matching decisive for simulation performance?
- What is easier to read and write?

$$A(x) + A(x+1) \longrightarrow \emptyset \qquad A(x) + A(y) \xrightarrow[y=x+1]{} \emptyset$$