On Matching Concurrent Traces

Iliano Cervesato Frank Pfenning Jorge Luis Sacchini
Carsten Schürmann Robert J. Simmons

July 1, 2012

Concurrent Traces

- A concurrent trace is a sequence of computational steps where independent steps can be permuted
- Computational traces define a model for concurrent and distributed computations
- We analyze the matching problem on concurrent traces

Long-term objectives

- Develop a logical framework to reason about concurrent traces
- Unification and matching are prerequisites
- This work: matching on concurrent traces, defined by multiset rewriting system, with one variable standing for an unknown trace

Multiset Rewriting Systems

- A multiset rewriting system is a set of rules of the form

$$r : \tilde{a} \rightarrow \tilde{b}$$

  where $\tilde{a}$ and $\tilde{b}$ are multisets of elements of S (support set)
- Rules represent state transformations
- A state is a set of pairs of the form $x : a$ where x is a unique name, and $a \in S$
- Applying a rule is represented by

$$R \vdash s \xrightarrow{t} s'$$

$$R, r : \tilde{a} \rightarrow \tilde{b} \vdash s, \tilde{x} : \tilde{a} \xrightarrow{r(\tilde{x},\tilde{y})} s, \tilde{y} : \tilde{b}$$

Concurrent traces

- A trace is a sequence of computational steps

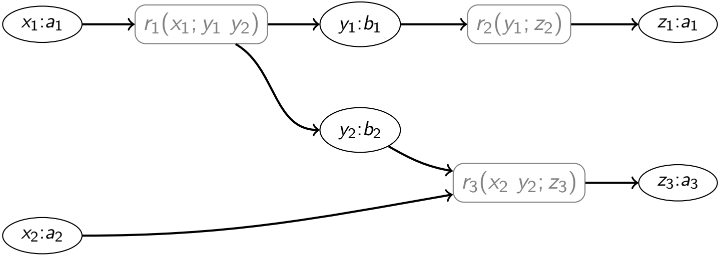
$$t ::= \cdot \mid r(\tilde{x}, \tilde{y}) \mid t_1; t_2$$

- Multistep computation

$$R \vdash s \stackrel{t}{\Longrightarrow} s'$$

Concurrent traces

- Example



$$x_1{:}a_1, x_2{:}a_2 \xRightarrow{r_1(x_1,y_1\ \ y_2);r_2(y_1,z_1);r_3(y_2\ \ x_2,z_2)} z_1{:}a_1, z_3{:}a_3$$

- Independent steps can be permuted

Concurrent traces

- Pre- and post-conditions:

$$\begin{cases} \bullet(\cdot) & = & \cdot \\ \bullet r(\tilde{x};\tilde{y}) & = & \tilde{x} : \tilde{a} \\ \bullet(\vec{t_1};\vec{t_2}) & = & \bullet\vec{t_1} \cup (\bullet\vec{t_2} \setminus \vec{t_1}\bullet) \end{cases}$$

$$\begin{cases} (\cdot)\bullet & = & \cdot \\ r(\tilde{x};\tilde{y})\bullet & = & \tilde{y} : \tilde{b} \\ (\vec{t_1};\vec{t_2})\bullet & = & (\vec{t_1}\bullet \setminus \bullet\vec{t_2}) \cup \vec{t_2}\bullet \end{cases}$$

Concurrent traces

- Trace independence

$$t_1 \parallel t_2 \iff \bullet t_1 \cap t_2 \bullet = t_1 \bullet \cap \bullet t_2 = \emptyset$$

- Trace equality: $t = t'$

$$R \vdash s \xRightarrow{t} s' \qquad \iff \qquad R \vdash s \xRightarrow{t'} s'$$

- Equations

$$\begin{aligned}
(\vec{t_1}; \vec{t_2}); \vec{t_3} &= \vec{t_1}; (\vec{t_2}; \vec{t_3}) \\
\cdot; \vec{t} &= \vec{t} \\
\vec{t}; \cdot &= \vec{t} \\
\vec{t_1}; \vec{t_2} &= \vec{t_2}; \vec{t_1} & \text{if } \vec{t_1} \parallel \vec{t_2}
\end{aligned}$$

Concurrent traces

- Trace equality is given by the binding structure
- Internal bindings can be renamed

$$p(x; y); q(y; z) \quad = \quad p(x; y'); q(y'; z)$$

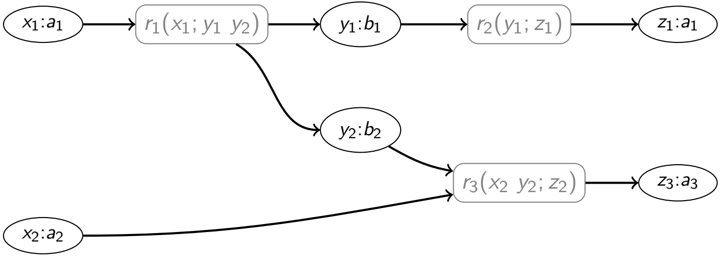
but

$$p(x; y); q(y; z) \quad \neq \quad p(x'; y); q(y; z')$$

- A renaming is legal if it is the identity for $\bullet t$ and $t\bullet$

Concurrent traces

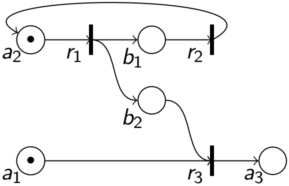
- Example



$$r_1(x_1, y_1 \;\; y_2); r_2(y_1, z_1); r_3(y_2 \;\; x_2, z_2)$$
$$=$$
$$r_1(x_1, y_1 \;\; y_2); r_3(y_2 \;\; x_2, z_3); r_2(y_1, z_1)$$

Concurrent traces

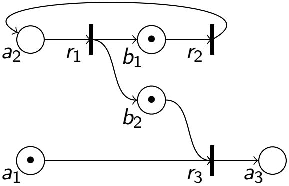
Concurrent traces are equivalent to executions on Petri nets



$$s = x_1{:}a_1, x_2{:}a_2$$

Concurrent traces

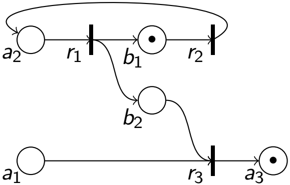
Concurrent traces are equivalent to executions on Petri nets



$$s = x_1{:}a_1, x_2{:}a_2 \xrightarrow{r_1(x_2,y_1\ \ y_2)} x_1{:}a_1, y_1{:}b_1, y_2{:}b_2$$

Concurrent traces

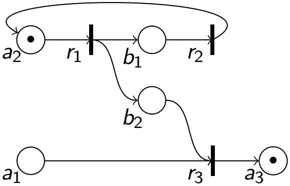
Concurrent traces are equivalent to executions on Petri nets



$$
\begin{aligned}
s = x_1{:}a_1, x_2{:}a_2 &\xrightarrow{r_1(x_2,y_1\ \ y_2)} x_1{:}a_1, y_1{:}b_1, y_2{:}b_2 \\
&\xrightarrow{r_3(x_1\ \ y_2,z_3)} y_1{:}b_1, y_2{:}b_2, z_3{:}a_3
\end{aligned}
$$

Concurrent traces

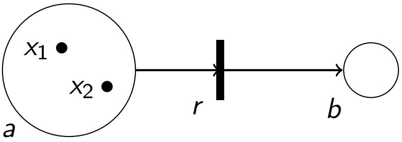
Concurrent traces are equivalent to executions on Petri nets



$$
\begin{aligned}
s = x_1{:}a_1, x_2{:}a_2 &\xrightarrow{r_1(x_2,y_1\ \ y_2)} x_1{:}a_1, y_1{:}b_1, y_2{:}b_2 \\
&\xrightarrow{r_3(x_1\ \ y_2,z_3)} y_1{:}b_1, y_2{:}b_2, z_3{:}a_3 \\
&\xrightarrow{r_2(y_1,z_2)} z_2{:}a_2, z_3{:}a_3
\end{aligned}
$$

Concurrent traces

- Tokens are named in our representation



$$x_1{:}a, x_2{:}a \xrightarrow{r(x_1,y)} y{:}b, x_2{:}a$$

$$x_1{:}a, x_2{:}a \xrightarrow{r(x_2,y)} x_1{:}a, y{:}b$$

Equations over traces

- Trace variables

$$t ::= \cdot \mid r(\tilde{x}; \tilde{y}) \mid t_1; t_2 \mid {\color{red} X(\tilde{x}; \tilde{y})}$$

- $\tilde{x}$ and $\tilde{y}$ represent the input and output interface of X
- X can be instantiated with a trace t such that $\bullet t = \tilde{x}$ and $t\bullet = \tilde{y}$
- An equation is given by a pair of traces containing variables

$$\vec{t_1} \stackrel{?}{=} \vec{t_2}$$

Matching on traces

- Given
$$t_1 \stackrel{?}{=} t_2 \qquad\qquad \text{with } t_2 \text{ ground}$$
Find a substitution $\theta = X_1 \leftarrow t_1, \ldots, X_n \leftarrow t_n$ such that
$$t_1[\theta] = t_2$$

Matching on traces

- Given

$$t_1 \stackrel{?}{=} t_2 \qquad\qquad \text{with } t_2 \text{ ground}$$

  Find a substitution $\theta = X_1 \leftarrow t_1, \ldots, X_n \leftarrow t_n$ such that

$$t_1[\theta] = t_2$$

- Matching is inherently non-deterministic: the following problem encodes multiset matching

$$X(\cdot;\cdot); Y(\cdot;\cdot) \stackrel{?}{=} c_1(\cdot;\cdot); \ldots; c_n(\cdot;\cdot)$$

  Partition $\{c_1, \ldots, c_n\}$ in two sets X and Y.

Matching on traces

- Matching is decidable:

$$t_1; X(\tilde{x}, \tilde{y}); t_1' \quad \stackrel{?}{=} \quad t_2$$

  $X(\tilde{x}, \tilde{y})$ must be instantiated with a subtrace of t_2

- In a problem

$$t_1^1; X_1(\tilde{x_1}, \tilde{y_1}); t_2^1; \ldots X_n(\tilde{x_n}, \tilde{y_n}); t_n^1 \quad \stackrel{?}{=} \quad t_2$$

  we can try all possible combinations of subtraces of t_2

Matching on traces

- Matching is decidable:

$$t_1; X(\tilde{x}, \tilde{y}); t_1' \quad \stackrel{?}{=} \quad t_2$$

  $X(\tilde{x}, \tilde{y})$ must be instantiated with a subtrace of t_2

- In a problem

$$t_1^1; X_1(\tilde{x_1}, \tilde{y_1}); t_2^1; \ldots X_n(\tilde{x_n}, \tilde{y_n}); t_n^1 \quad \stackrel{?}{=} \quad t_2$$

  we can try all possible combinations of subtraces of t_2

- Expensive

Matching on traces

- We consider 1-variable matching

$$t_1; X(\tilde{x_1}, \tilde{y_1}); t_1' \quad \stackrel{?}{=} \quad t_2$$

where t_1, t_1' and t_2 are ground

Matching on traces

- We consider 1-variable matching

$$t_1; X(\tilde{x_1}, \tilde{y_1}); t_1' \quad \stackrel{?}{=} \quad t_2$$

  where t_1, t_1' and t_2 are ground
- Reduces search space

Intuition of the algorithm

- Match individual computation steps until we are left with the logic variable

$$t_1; X(\tilde{x}_1; \tilde{y}_1); t_2 \quad \stackrel{?}{=} \quad u_1$$

Intuition of the algorithm

- Match individual computation steps until we are left with the logic variable

$$t_1; X(\tilde{x}_1; \tilde{y}_1); t_2 \quad \stackrel{?}{=} \quad p(\tilde{x}; \tilde{y}); u_2$$

Intuition of the algorithm

- Match individual computation steps until we are left with the logic variable

$$p(\tilde{x};\tilde{y}); t_3; X(\tilde{x_1};\tilde{y_1}); t_2 \quad \stackrel{?}{=} \quad p(\tilde{x};\tilde{y}); u_2$$

Intuition of the algorithm

- Match individual computation steps until we are left with the logic variable

$$t_3; X(\tilde{x}_1; \tilde{y}_1); t_2 \quad \stackrel{?}{=} \quad u_2$$

Intuition of the algorithm

- Match individual computation steps until we are left with the logic variable

$$\begin{array}{rcl} t_3; X(\tilde{x}_1; \tilde{y}_1); t_2 & \stackrel{?}{=} & u_2 \\ & \vdots & \\ X(\tilde{x}_1; \tilde{y}_1); t_4 & \stackrel{?}{=} & u_3 \end{array}$$

Intuition of the algorithm

- Match individual computation steps until we are left with the logic variable

$$
\begin{array}{rcl}
t_3; X(\tilde{x}_1; \tilde{y}_1); t_2 & \stackrel{?}{=} & u_2 \\
& \vdots & \\
X(\tilde{x}_1; \tilde{y}_1); t_4 & \stackrel{?}{=} & u_4; p'(\tilde{x}_2; \tilde{y}_2)
\end{array}
$$

Intuition of the algorithm

- Match individual computation steps until we are left with the logic variable

$$\begin{array}{rcl} t_3; X(\tilde{x}_1; \tilde{y}_1); t_2 & \stackrel{?}{=} & u_2 \\ & \vdots & \\ X(\tilde{x}_1; \tilde{y}_1); t_5; p'(\tilde{x}_2; \tilde{y}_2) & \stackrel{?}{=} & u_4; p'(\tilde{x}_2; \tilde{y}_2) \end{array}$$

Intuition of the algorithm

- Match individual computation steps until we are left with the logic variable

$$
\begin{array}{rcl}
t_3; X(\tilde{x_1}; \tilde{y_1}); t_2 & \stackrel{?}{=} & u_2 \\
 & \vdots & \\
X(\tilde{x_1}; \tilde{y_1}); t_5 & \stackrel{?}{=} & u_4
\end{array}
$$

Intuition of the algorithm

- Match individual computation steps until we are left with the logic variable

$$
\begin{array}{rcl}
t_3; X(\tilde{x}_1; \tilde{y}_1); t_2 & \stackrel{?}{=} & u_2 \\
& \vdots & \\
X(\tilde{x}_1; \tilde{y}_1); t_5 & \stackrel{?}{=} & u_4 \\
& \vdots & \\
X(\tilde{x}_1; \tilde{y}_1) & \stackrel{?}{=} & u_5
\end{array}
$$

Intuition of the algorithm

- Match individual computation steps until we are left with the logic variable

$$\begin{array}{rcl} t_3; X(\tilde{x}_1; \tilde{y}_1); t_2 & \stackrel{?}{=} & u_2 \\ & \vdots & \\ X(\tilde{x}_1; \tilde{y}_1); t_5 & \stackrel{?}{=} & u_4 \\ & \vdots & \\ X(\tilde{x}_1; \tilde{y}_1) & \stackrel{?}{=} & u_5 \end{array}$$

Solution: $X \leftarrow u_5$

Algorithm

1. $p(\tilde{x};\tilde{y});\vec{t_1} \stackrel{?}{=} p(\tilde{x};\tilde{y}');\vec{t_2}$:
   If $\tilde{y}'/\tilde{y}$ is legal for $p(\tilde{x};\tilde{y});\vec{t_1}$, then solve $[\tilde{y}'/\tilde{y}]\vec{t_1} \stackrel{?}{=} \vec{t_2}$, otherwise fail.
2. $\vec{t_1};p(\tilde{x};\tilde{y}) \stackrel{?}{=} \vec{t_2};p(\tilde{x}';\tilde{y})$:
   If $\tilde{x}'/\tilde{x}$ is legal for $\vec{t_1};p(\tilde{x};\tilde{y})$, then solve $[\tilde{x}'/\tilde{x}]\vec{t_1} \stackrel{?}{=} \vec{t_2}$, otherwise fail.
3. $X(\tilde{x};\tilde{y}) \stackrel{?}{=} \vec{t_2}$: Simply return the solution $X \leftarrow \vec{t_2}$.

Invariant: in $t_1 \stackrel{?}{=} t_2$, $\bullet t_1 = \bullet t_2$ and $t_1\bullet = t_2\bullet$

Example

- Match failure:

$$p(x_1; y_1\ \ y_2); X(y_1\ \ y_2; z_1\ \ z_2) \quad \stackrel{?}{=} \quad p(x_1; y_1'\ \ z_2); q(y_1'; z_1)$$

  Matching fails because there is no legal renaming between

$$y_1\ \ y_2 \longrightarrow y_1'\ \ z_2$$

Correctness of the algorithm

Soundness

Given a matching problem $\vec{t_1}; X(\tilde{x}; \tilde{y}); \vec{t'_1} \stackrel{?}{=} \vec{t_2}$, if the matching algorithm reports a solution $X \leftarrow \vec{t}$, then there is a legal renaming ρ such that $\rho\vec{t_1}; \vec{t}; \rho\vec{t'_1} = \vec{t_2}$.

Proof

By induction on the size of the trace.

Correctness of the algorithm

Completeness

Given a matching problem $\vec{t_1}; X(\tilde{x}; \tilde{y}); \vec{t'_1} \stackrel{?}{=} \vec{t_2}$, if there is a trace $\vec{t}$ such that $\vec{t_1}; \vec{t}; \vec{t'_1} = \vec{t_2}$, then the matching algorithm will report the solution $X \leftarrow \rho\vec{t}$ for some renaming ρ.

Proof

By induction on the size of the trace.

Extensions

- Extend matching to CLF
  - Affine and persistent functions
  - Dependent types
  - This work: propositional linear fragment of CLF
- Current results
  - 1-var matching for larger fragments CLF
  - 1-var unification for simply-typed fragment of CLF:

$$t_1; X_1; t_1' \stackrel{?}{=} t_2; X_2; t_2'$$

Conclusions

- We presented an algorithm for 1-var matching on concurrent traces based on multiset rewrite systems
- These results have been extended to larger systems (fragments of CLF)
- Future work
  - Same variable on both sides: $X[x\ y, \cdot] \stackrel{?}{=} X[y\ x, \cdot]$
  - n-var unification
- Long-term objectives
  - Reason about concurrent traces

Conclusions

- We presented an algorithm for 1-var matching on concurrent traces based on multiset rewrite systems
- These results have been extended to larger systems (fragments of CLF)
- Future work
  - Same variable on both sides: $X[x\ y, \cdot] \stackrel{?}{=} X[y\ x, \cdot]$
  - n-var unification
- Long-term objectives
  - Reason about concurrent traces

Thank you!