A Choreographic Language for PRISM

... Author: Please enter affiliation as second parameter of the author macro

... Author: Please enter affiliation as second parameter of the author macro

Abstract

This is the abstract

2012 ACM Subject Classification Theory of computation → Type theory; Computing methodologies → Distributed programming languages; Theory of computation → Program verification

Keywords and phrases Session types, PRISM, Model Checking

Digital Object Identifier 10.4230/LIPIcs.ITP.2023.m

Funding This work was supported by

1 Formal Language

In this section, we provide the formal definition of our choreographic language as well as process algebra representing PRISM [**?**].

1.1 Choreographies

Syntax. Our choreographic language is defined by the following syntax:

$$
\begin{array}{llll}
(\text{Chor}) & C & ::= & \{\mathsf{p}_i\}_{i\in I} + \{\lambda_j : x_j = E_j;\ C_j\}_{j\in J} \mid \mathsf{if}\ E@\mathsf{p}\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \mid X \mid \mathbf{0} \\
(\text{Expr}) & E & ::= & f(\tilde{E}) \quad \mid \quad x \quad \mid \quad v \\
(\text{Rates}) & \lambda \in \mathbb{R} & & (\text{Variables}) \quad x \in \mathsf{Var} \qquad (\text{Values}) \quad v \in \mathsf{Val}
\end{array}
$$

We briefly comment the various constructs. The syntactic category C denotes choreographic programmes. The term $\mathsf{p} \longrightarrow \{\mathsf{p}_i\}_{i\in I} \oplus \{[\lambda_j]x_j = E_j : C_j\}_{j\in J}$ denotes an interaction between roles p_i...

1.2 PRISM

Syntax.

$$
\begin{array}{llcll}
(\text{Networks}) & N, M & ::= & \mathbf{0} & \text{empty network} \\
 & & \mid & \mathsf{p} : \{F_i\}_i & \text{module} \\
 & & \mid & M|[A]|M & \text{parallel composition} \\
 & & \mid & M/A & \text{action hiding} \\
 & & \mid & \sigma M & \text{substitution} \\
 & & & & \\
(\text{Commands}) & F & ::= & [a]g \to \Sigma_{i\in I}\{\lambda_i : u_i\} & g \text{ is a boolean expression in } E \\
 & & & & \\
(\text{Assignment}) & u & ::= & (x' = E) & \text{update } x\text{, element of } \mathcal{V}\text{, with } E \\
 & & \mid & A\&A & \text{multiple assignments}
\end{array}
$$

Semantics. We construct all the enables commands by applying a closure to the following

International Conference on Blah.
Editors: John Q. Open and Joan R. Access; Article No. m; pp. m:1–m:6
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

rules.

$$\frac{[\,]E \to \{\lambda_i : x_i = E_i\}_{i\in I} \in \{\![M_j]\!\} \quad j \in \{1,2\}}{[\,]E \to \{\lambda_i : x_i = E_i\}_{i\in I} \in \{\![M_1|[A]|M_2]\!\}}$$

$$\frac{[a]E \to \{\lambda_i : x_i = E_i\}_{i\in I} \in \{\![M_j]\!\} \quad a \notin A \quad j \in \{1,2\}}{[a]E \to \{\lambda_i : x_i = E_i\}_{i\in I} \in \{\![M_1|[A]|M_2]\!\}}$$

$$\frac{[a]E \to \{\lambda_j : x_i = E_i\}_{i\in I} \in \{\![M_1]\!\} \quad [a]E' \to \{\lambda_j : x'_j = E'_j\}_{j\in J} \in \{\![M_2]\!\} \quad a \in A}{[a]E \wedge E' \to \{\lambda_i * \lambda'_j : x_i = E_i \wedge x'_j = E'_j\}_{i\in I, j\in J} \in \{\![M_1|[A]|M_2]\!\}}$$

That means that ones we have a set of executable rules, we can start building a transition system. In order to do so, we

$$W(M) = \{F \mid F \in \{\![M]\!\}\}$$

$$X = \{x_1, \ldots, x_n\}$$

$$\sigma : X \to V$$

1.3 Projection from Choreographies to PRISM

Mapping Choreographies to PRISM. We need to run some standard static checks because, since there is branching, some terms may not be projectable.

$f : C \to \mathcal{R} \mapsto F$

$$f(\mathsf{p}_1 \longrightarrow \{\mathsf{p}_i\}_{i\in I} \oplus \{[\lambda_j]x_j = E_j : C_j\}_{j\in J}) =$$
$$= \begin{cases} \left([\lambda_{j_1}]x_{j_1} = f(E_{j_1})\right)_{\mathsf{p}} . f(\oplus\{[\lambda_j]x_j = E_j : C_j\}_{j\in J\setminus\{j_1\}}).f(C_{j_1}) & if\ \mathsf{p} = \mathsf{p}_1 \vee \mathsf{p} \in \{\mathsf{p}_i\}_{i\in I} \\ f(C_j) & if\ \mathsf{p} \neq \mathsf{p}_1 \wedge \mathsf{p} \notin \{\mathsf{p}_i\}_{i\in I} \end{cases}$$

$$f(\mathsf{if}\ E@\mathsf{p}\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2) = \begin{cases} f(E).f(C_1).f(C_2) & if\ \mathsf{p} \in \mathbf{roles} \\ \perp & otherwise \end{cases}$$

$$f(X) = ??$$
$$f(\mathbf{0}) = \perp$$

$f : [C_1, \ldots, C_n] \to String \mapsto String$

```
CASE 1: C_i ≡ p_1 ⟶ {p_i}_{i∈I} ⊕ {[λ_j]x_j = E_j : C_j}_{j∈J}

  f([C_i,...,C_n], code) :
     label = generateNewLabel()
     for c_j in C_i :
        for p ∈ roles(C_i):
           newCode = "[label] (x_j = E_j)_p"
           code = code + newCode
     f([C_{i+1},...,C_n],code)
```

where $\mathbf{roles}(\mathrm{C}_i) \coloneqq \mathsf{p}_1 \cup \{\mathsf{p}_i\}_{i\in I}$.

```
CASE 2: C_i ≡ if E@p then C_1 else C_2

    f([C_i,...,C_n], code) :
        code = code + (E)_p
        f(C_1,code)
        f(C_2,code)
        f([C_{i+1},...,C_n],code)
```

$\texttt{network} : \mathcal{R} \longrightarrow \texttt{Set}(F)$

$\texttt{f}(C_1, \ldots, C_n, \texttt{network}) =$

$\texttt{CASE 1:}\ \forall i.C_i \quad \equiv \quad \mathsf{p}_1 \longrightarrow \{p_i\}_{i \in I} \oplus \{[\lambda_j]x_j = E_j : D_j\}_{j \in J}$

```
->

    label = generateNewLabel()




    for c_j in C_i :
    for p ∈ roles(C_i):
    newCode = "[label] (x_j = E_j)_p"
    code = code + newCode
    f([C_{i+1},...,C_n],code)
```

$$f\Big(\quad \mathsf{p}_1 \longrightarrow \{\mathsf{p}_2\} \oplus \left\{ \begin{array}{l} [\lambda_1]x = 5 : \mathsf{p}_1 \longrightarrow \{\mathsf{p}_2\} \oplus \{[\lambda_3]y = 5\} \\ [\lambda_2]y = 10 : \mathsf{p}_1 \longrightarrow \{\mathsf{p}_2\} \oplus \{[\lambda_4]x = 10\} \end{array} \right\},\ \mathsf{p}_1 : \emptyset \parallel \mathsf{p}_2 : \emptyset\Big)$$

$$=$$

$$\begin{array}{l}
\text{label} = \text{newlabel}(); \\
\text{for } \mathsf{p}_i\{ \\
\quad add(p_i, [\text{label}]s_{p_i} = state(p_i) \rightarrow \left\{ \begin{array}{l} \lambda_1 : x' = 5; \mathsf{state}(\mathsf{p}_i)' = \mathsf{generatenewstate}(\mathsf{p}_i) \\ \lambda_2 : y' = 10; \mathsf{state}(\mathsf{p}_i)' = \mathsf{generatenewstate}(\mathsf{p}_i) \end{array} \right\} \\
\quad f(\mathsf{p}_1 \longrightarrow \{\mathsf{p}_2\} \oplus \{[\lambda_3]y = 5\}, network') = network'' \\
\quad return f(\mathsf{p}_1 \longrightarrow \{\mathsf{p}_2\} \oplus \{[\lambda_4]x = 10\}, network'')
\end{array}$$

$f : C \longrightarrow \texttt{network} \longrightarrow \texttt{network} \qquad \texttt{network} : \mathcal{R} \longrightarrow \text{Set}(F)$

$$f\Big(\ \mathsf{p}_1 \longrightarrow \{\mathsf{p}_i\}_{i \in I} \oplus \{[\lambda_j]x_j = E_j : D_j\}_{j \in J}, \texttt{network}\Big)$$

$$=$$

$$\begin{array}{l}
\mathsf{label} = \texttt{newlabel}(); \\
\mathsf{for}\ \mathsf{p}_k \in \textbf{roles}\{ \\
\quad \mathsf{for}\ j \in J\{ \\
\quad\quad \texttt{network} = \texttt{add}(\mathsf{p}_k, [\mathsf{label}]s_{\mathsf{p}_k} = \texttt{state}(\mathsf{p}_k) \rightarrow\ \lambda_j : x_j = E_j\ \&\ s'_{\mathsf{p}_k} = \mathsf{genNewState}(\mathsf{p}_k)); \\
\quad \} \\
\} \\
\mathsf{for}\ j \in J\{ \\
\quad \texttt{network} = f(D_j, \texttt{network}); \\
\} \\
\mathsf{return}\ \texttt{network}
\end{array}$$

$$f\Big(\textsf{if } E@\mathsf{p} \textsf{ then } C_1 \textsf{ else } C_2, \texttt{network}\Big)$$

$$=$$

$$\begin{aligned}
&\texttt{network} = \texttt{add}(\mathsf{p}, [\,]s_{\mathsf{p}} = \texttt{state}(\mathsf{p})\ \&\ f(E));\\
&\texttt{network} = f(C_1, \texttt{network});\\
&\texttt{network} = f(C_2, \texttt{network});\\
&\text{return } \texttt{network}
\end{aligned}$$

2 Tests

Put tests/benchmarking here.

Each example should be described.

References