FAKULTA
INFORMATIKY
Masarykova univerzita

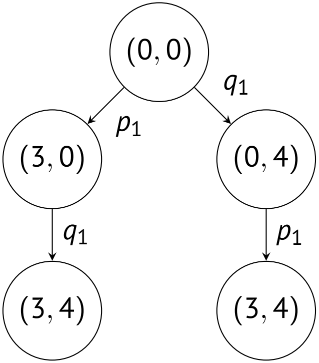
Context-Sensitive Dynamic Partial Order Reduction

Jan Tušil

26. března 2018

Example 1

```
p: x := 3     q: y := 4
```



$$\begin{tikzcd} & (0,0) \arrow[dl, "p_1"] \arrow[dr, "q_1"] & \\ (3,0) \arrow[d, "q_1"] & & (0,4) \arrow[d, "p_1"] \\ (3,4) & & (3,4) \end{tikzcd}$$

Example 2

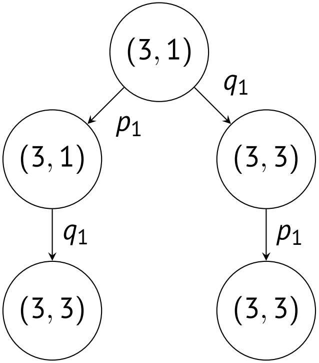
```
p: x := 3     q: y := x
```

$$\begin{tikzcd}
 & (0,0) \arrow[dl, "p_1"] \arrow[dr, "q_1"] & \\
(3,0) \arrow[d, "q_1"] & & (0,0) \arrow[d, "p_1"] \\
(3,3) & & (3,0)
\end{tikzcd}$$

FAKULTA INFORMATIKY
Masarykova univerzita

Example 3

```
p: x := 3     q: y := x
```



$$\begin{tikzcd}
 & (3,1) \arrow[dl, "p_1"] \arrow[dr, "q_1"] & \\
(3,1) \arrow[d, "q_1"] & & (3,3) \arrow[d, "p_1"] \\
(3,3) & & (3,3)
\end{tikzcd}$$

Pojmy

- Od naivního k Source-POR
- Od Source-POR k Context-Sensitive (Source-) POR

Notace

FAKULTA INFORMATIKY
Masarykova univerzita

Notace

val Event $\leftarrow$ Process $\times \mathbb{N}$

Notace

val Event $\leftarrow$ Process $\times\mathbb{N}$
val s : State

Notace

```
**val** Event ← Process $\times \mathbb{N}$
**val** s : State                                        ▷ Iniciální stav
```

Notace

```
**val** Event $\leftarrow$ Process $\times \mathbb{N}$
**val** s : State                                   ▷ Iniciální stav
**val** s[_] : Trace -> State
```

FAKULTA
INFORMATIKY
Masarykova univerzita

Notace

```
val Event ← Process × ℕ
val s : State                                   ▷ Iniciální stav
val s[_] : Trace -> State      ▷ Stav, do něhož doběhne stopa z s
```

FAKULTA
INFORMATIKY
Masarykova univerzita

Notace

val Event $\leftarrow$ Process $\times \mathbb{N}$
val s : State ▷ Iniciální stav
val s[_] : Trace -> State ▷ Stav, do něhož doběhne stopa z s
val enabled : State -> Set<Process>

Notace

```
**val** Event $\leftarrow$ Process $\times \mathbb{N}$
**val** s : State                                      ▷ Iniciální stav
**val** s[_] : Trace -> State         ▷ Stav, do něhož doběhne stopa z s
**val** enabled : State -> Set<Process>
**var** Sleep : Trace -> Set<Trace>
```

FAKULTA
INFORMATIKY
Masarykova univerzita

Notace

```
val Event ← Process ×ℕ
val s : State                                          ▷ Iniciální stav
val s[_] : Trace -> State           ▷ Stav, do něhož doběhne stopa z s
val enabled : State -> Set<Process>
var Sleep : Trace -> Set<Trace>   ▷ Navazující stopy jež netřeba řešit
```

Notace

```
**val** Event ← Process ×ℕ
**val** s : State                                      ▷ Iniciální stav
**val** s[_] : Trace -> State           ▷ Stav, do něhož doběhne stopa z s
**val** enabled : State -> Set<Process>
**var** Sleep : Trace -> Set<Trace>    ▷ Navazující stopy jež netřeba řešit
**var** backtrack : Trace -> Set<Process>
```

FAKULTA
INFORMATIKY
Masarykova univerzita

Notace

```
**val** Event $\leftarrow$ Process $\times \mathbb{N}$
**val** s : State                                   ▷ Iniciální stav
**val** s[_] : Trace -> State          ▷ Stav, do něhož doběhne stopa z s
**val** enabled : State -> Set<Process>
**var** Sleep : Trace -> Set<Trace>   ▷ Navazující stopy jež netřeba řešit
**var** backtrack : Trace -> Set<Process>
**val** dom : Trace -> Set<Event>
```

FAKULTA INFORMATIKY
Masarykova univerzita

Algoritmus

```
**function** ExploreCS($E$ : Trace, $Sleep$ : Set<Trace>)
    $sleep(E) \leftarrow Sleep$
    **choose process** $p \in enabled(s[e]) \setminus Sleep$ **or return**
    $backtrack(E) \leftarrow \{p\};$
    **while** $\exists p \in backtrack(E) \setminus sleep(E)$ **do**
        DetectRaces($E$, $p$)
        $Sleep' \leftarrow \{v \mid v \in sleep(E) \wedge E \models p \diamond v\}$
        $Sleep' \leftarrow Sleep' \cup \{v \mid p.v \in sleep(E)\}$
        Explore($E.p$, $Sleep'$)
        $sleep(E) \leftarrow sleep(E) \cup \{p\}$
    **end while**
**end function**
```

DetectRaces

```
**function** DetectRaces(E : Trace, p: Process)
    **val** $e_p$ : Event $\leftarrow$ $next_E(p)$
    **for all** $e \in dom(E)$ **such that** $e$ is in reversible race with $e_p$ **do**
        **val** $E' \leftarrow prefixBefore(E, e)$                    ▷ does not include $e$
        **val** $v$ : Trace $\leftarrow indepSuffixFrom(e, P).p$            ▷ includes $e$
        **if** the first event of $v$ is not in $backtrack(E')$ **then**
            add it there
        **end if**
    **end for**
**end function**
```

DetectRaces

```
function DetectRaces(E : Trace, p: Process)
    val $e_p$ : Event $\leftarrow next_E(p)$
    for all $e \in dom(E)$ such that $e$ is in reversible race with $e_p$ do
        val $E' \leftarrow prefixBefore(E, e)$                    $\triangleright$ does not include $e$
        val $v$ : Trace $\leftarrow indepSuffixFrom(e, P).p$              $\triangleright$ includes $e$
        if $I_{E'}(v) \cap backtrack(E') \neq \emptyset$ then
            add some $q' \in I_{E'}(v)$ to $backtrack(E')$
        end if
    end for
end function
```

Relace happened-before