Runtime Verification Based on Executable Models: On-the-Fly Matching of Timed Traces

Mikhail Chupilko,
Alexander Kamkin

Outline

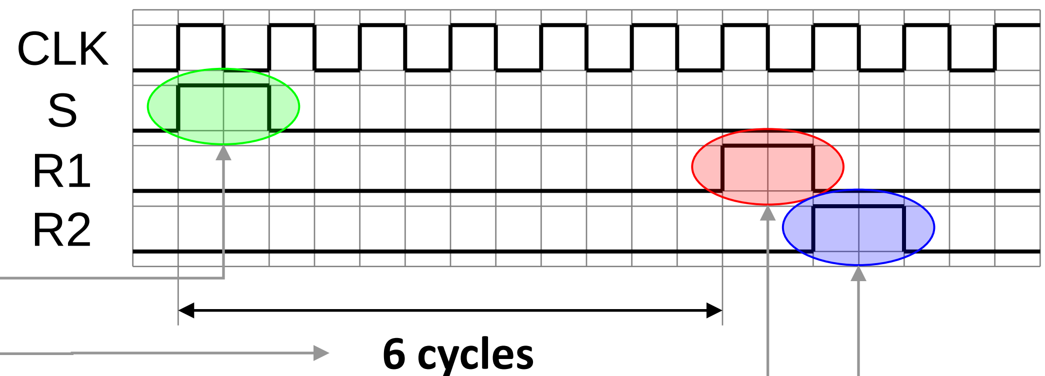
- Hardware models
- Runtime verification
- Elements of formalization
- Conformance relation
- Conclusion

Hardware models

- They are developed in Hardware Description Languages, like Verilog or VHDL
- The result of development is the program being executed in HDL simulator
- The common approach for verification of hardware models is testing of HDL programs
- To automatize testing is possible by means of executable models (e.g. in C++)

HDL programs

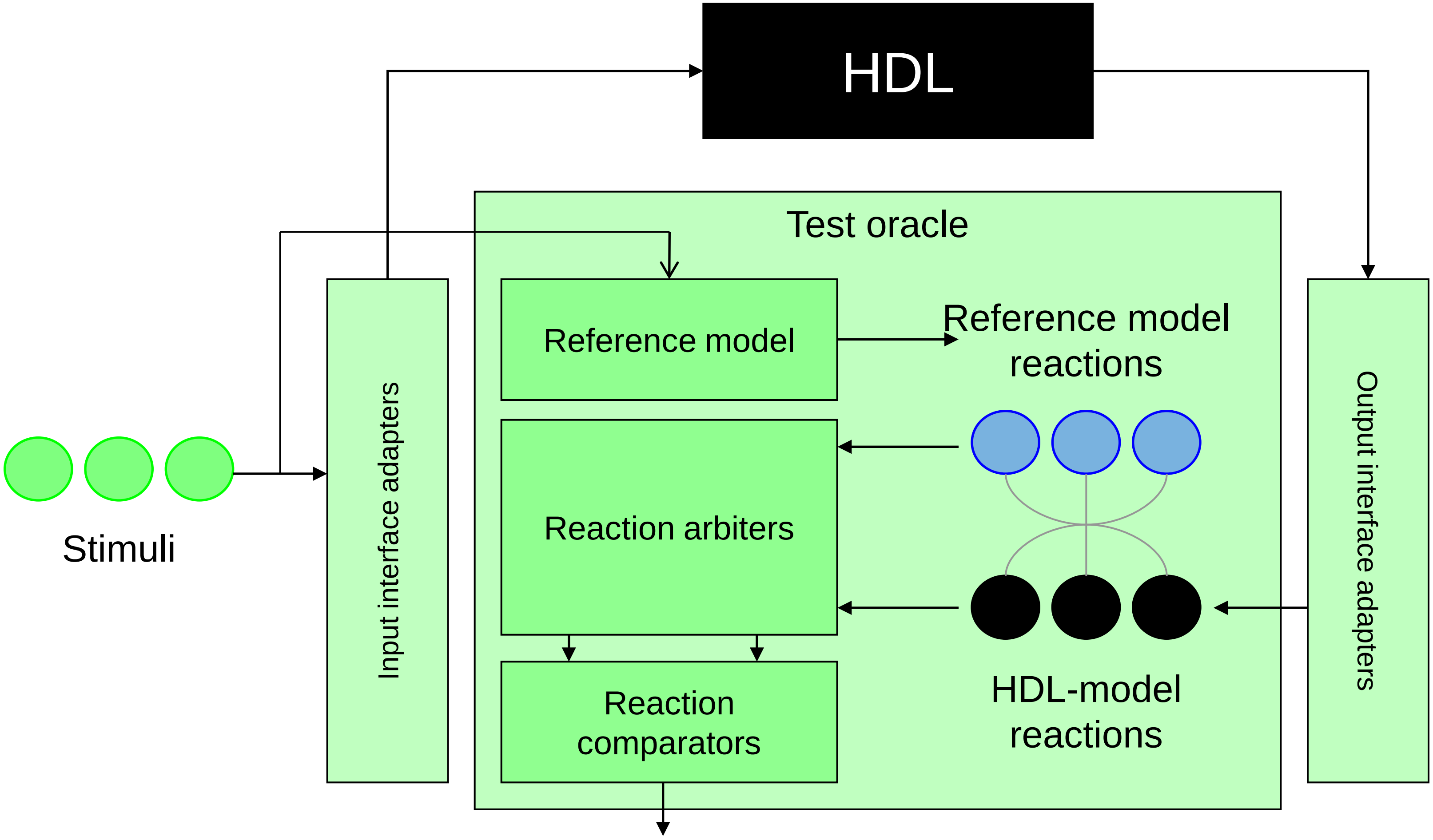
```
input S;
output R1, R2;
void design() {
    while(true) {
        wait(S);
        delay(6);
        R1 = 1;
        delay(1);
       {R1 = 0;
        R2 = 1;}
        delay(1);
        R2 = 0;
    }
}
```



Parallel assignments

Hardware model behavior

Reference model-based test oracle



Behavior correctness checking

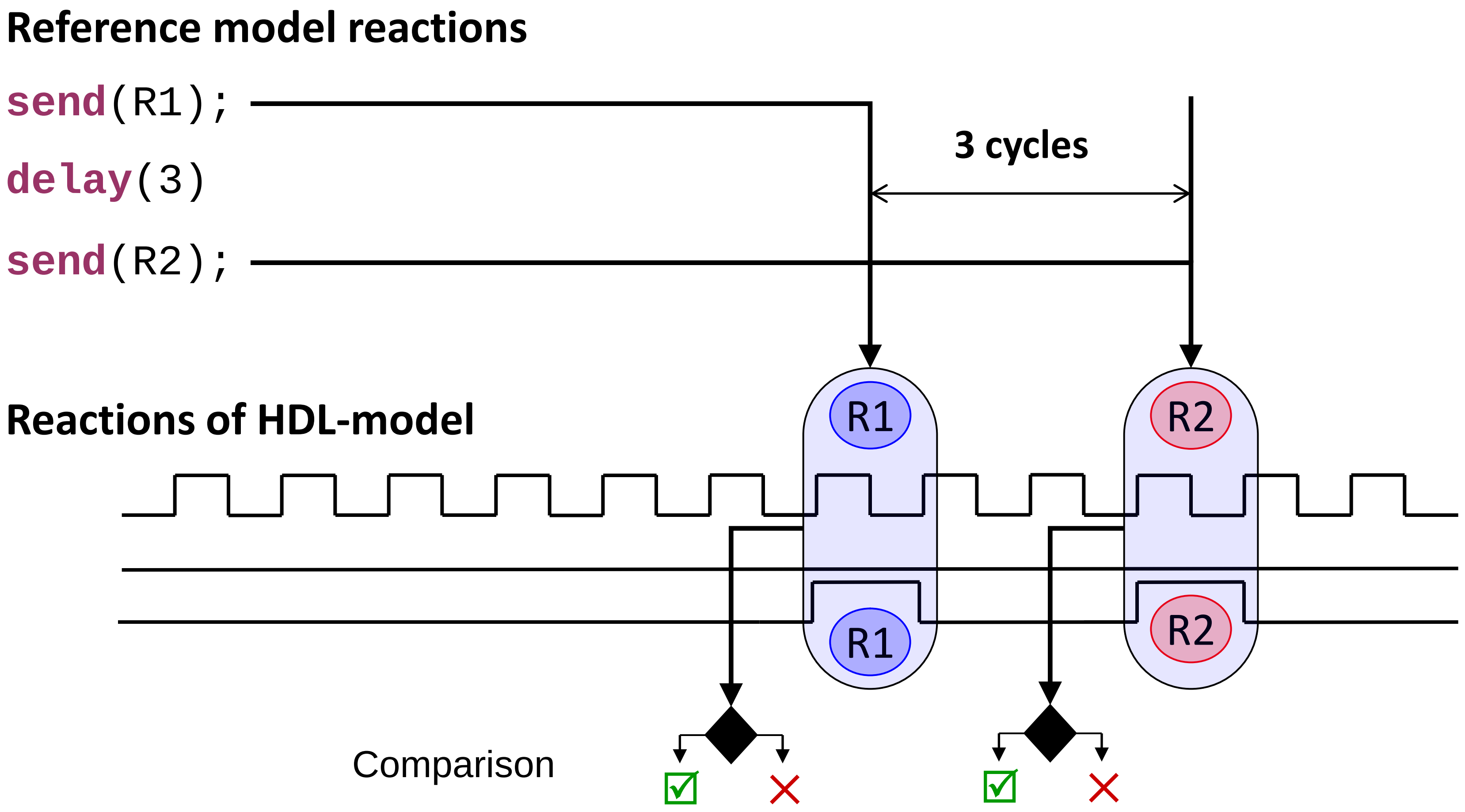
Functional properties

- Set of reactions is correct
- Each reaction is correct

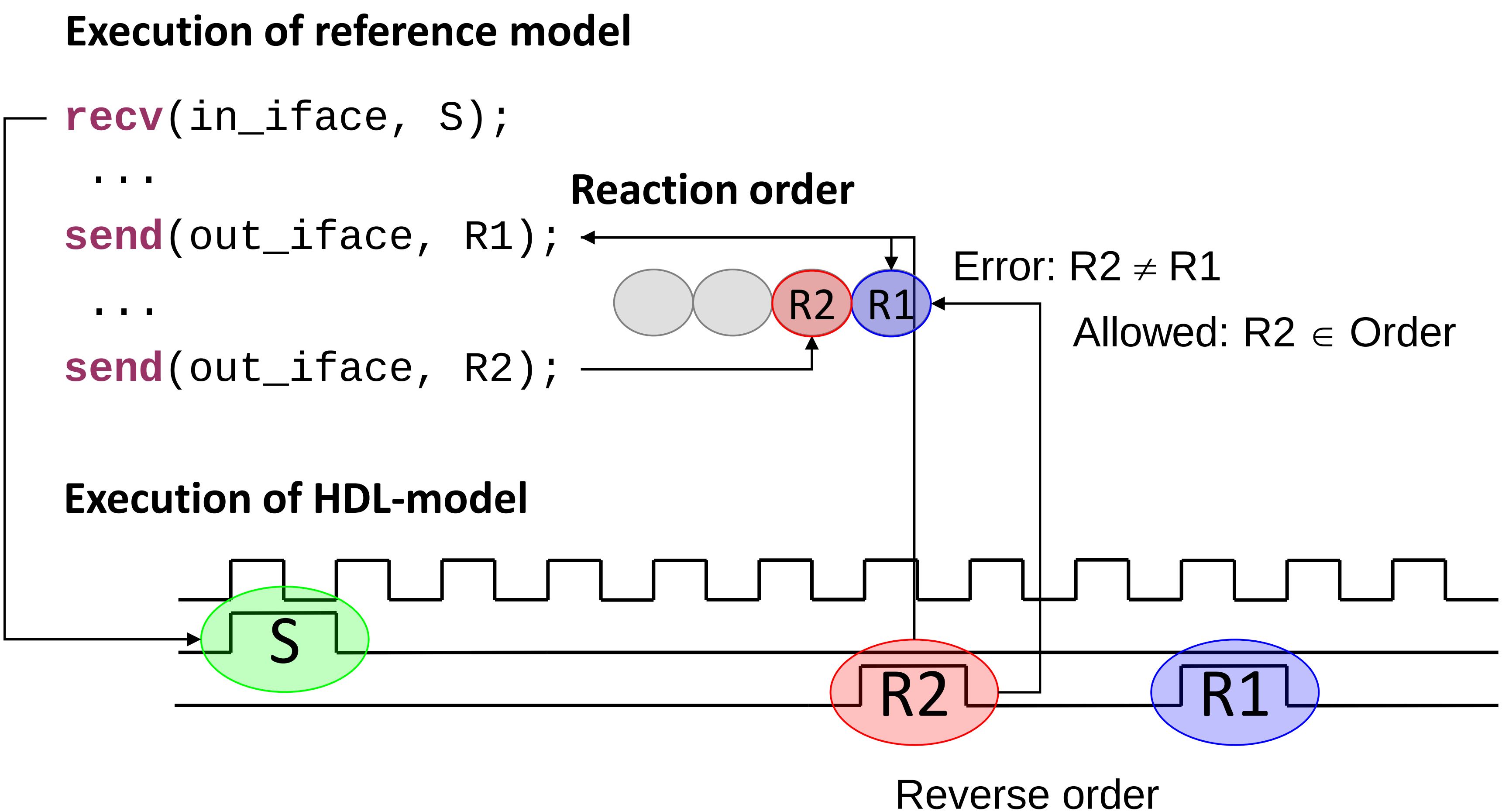
- Reaction order is correct
- Delays between reactions are correct

Time restrictions

Cycle-accurate checking



Ambiguity in reaction order



Arbitration of reactions

- Reaction arbiter finds a reaction corresponding to the reference model one
- Behavior checking depends on both reference model and on arbitration
- Reaction arbiters encapsulate parts of test oracle functionality aimed at reaction order checking

Types of reaction arbiters

- Deterministic model-based arbiter

  $arbiter$: $2^{Reaction} \to Reaction \cup \{fail\}$

- Adaptive arbiter

  $arbiter$: $2^{Reaction} \times Reaction \to Reaction \cup \{fail\}$

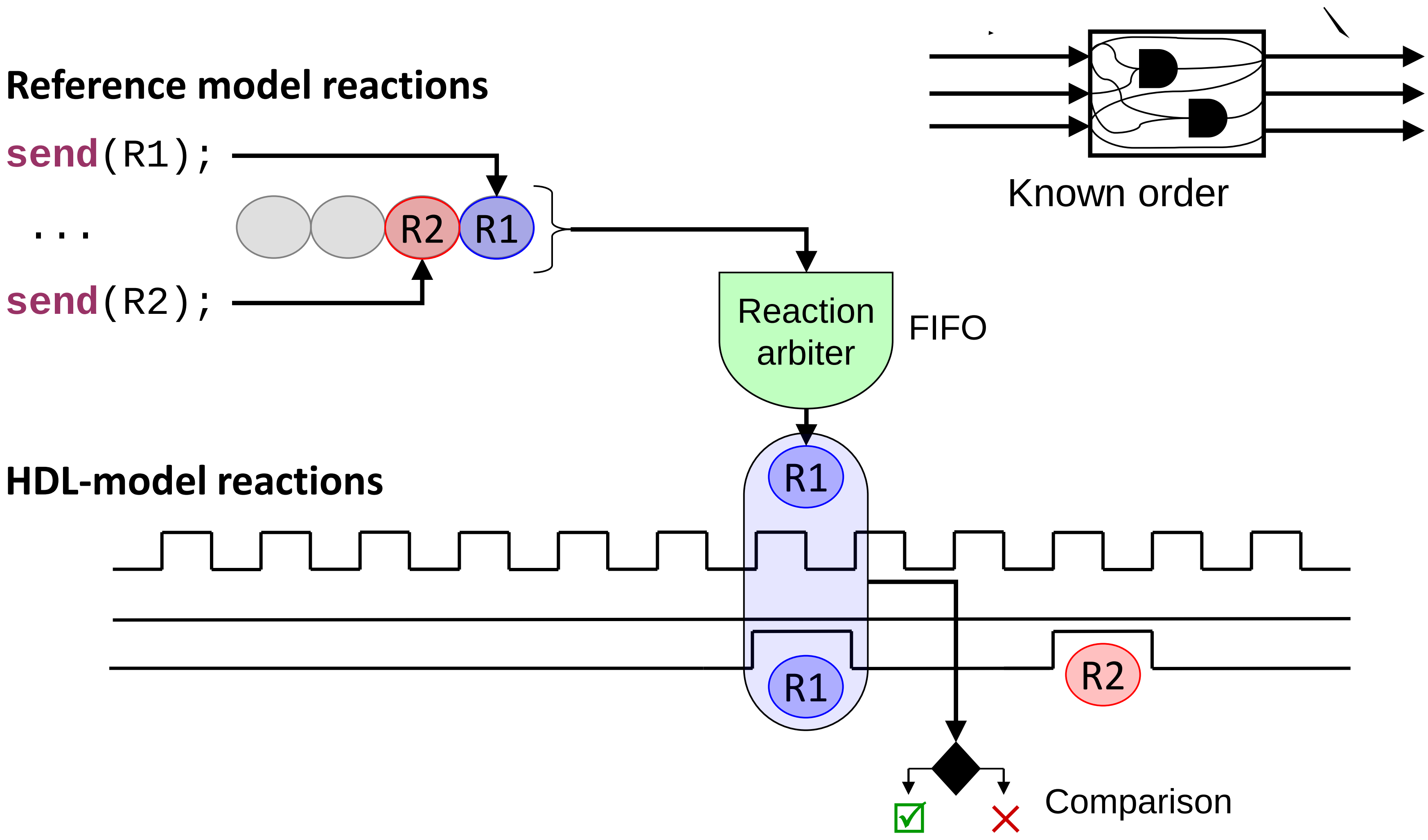
- Two-level arbiter

  $arbiter(reactions) \equiv arbiter_2(arbiter_1(reactions), reaction)$

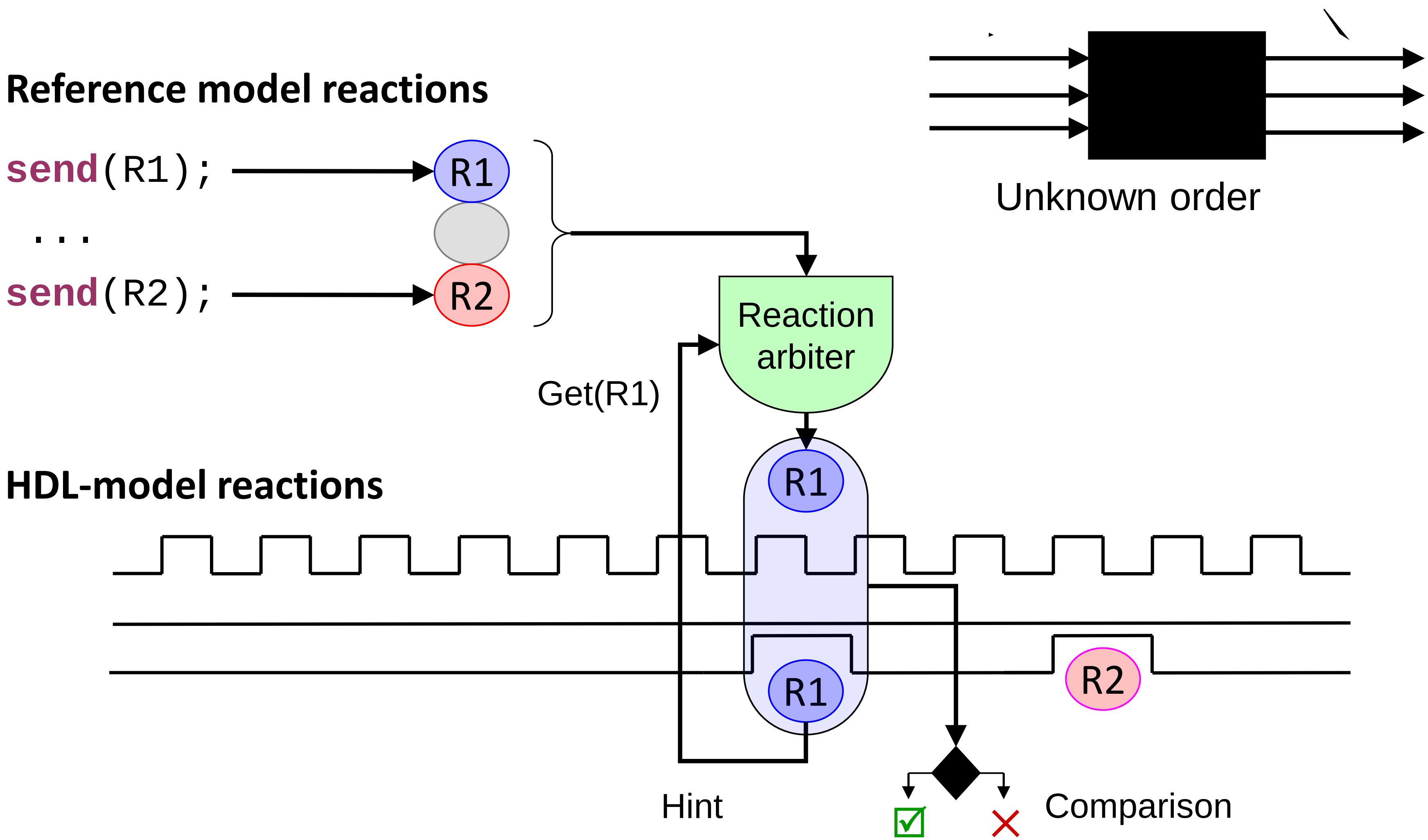
  – Non-deterministic arbiter

  – Adaptive arbiter

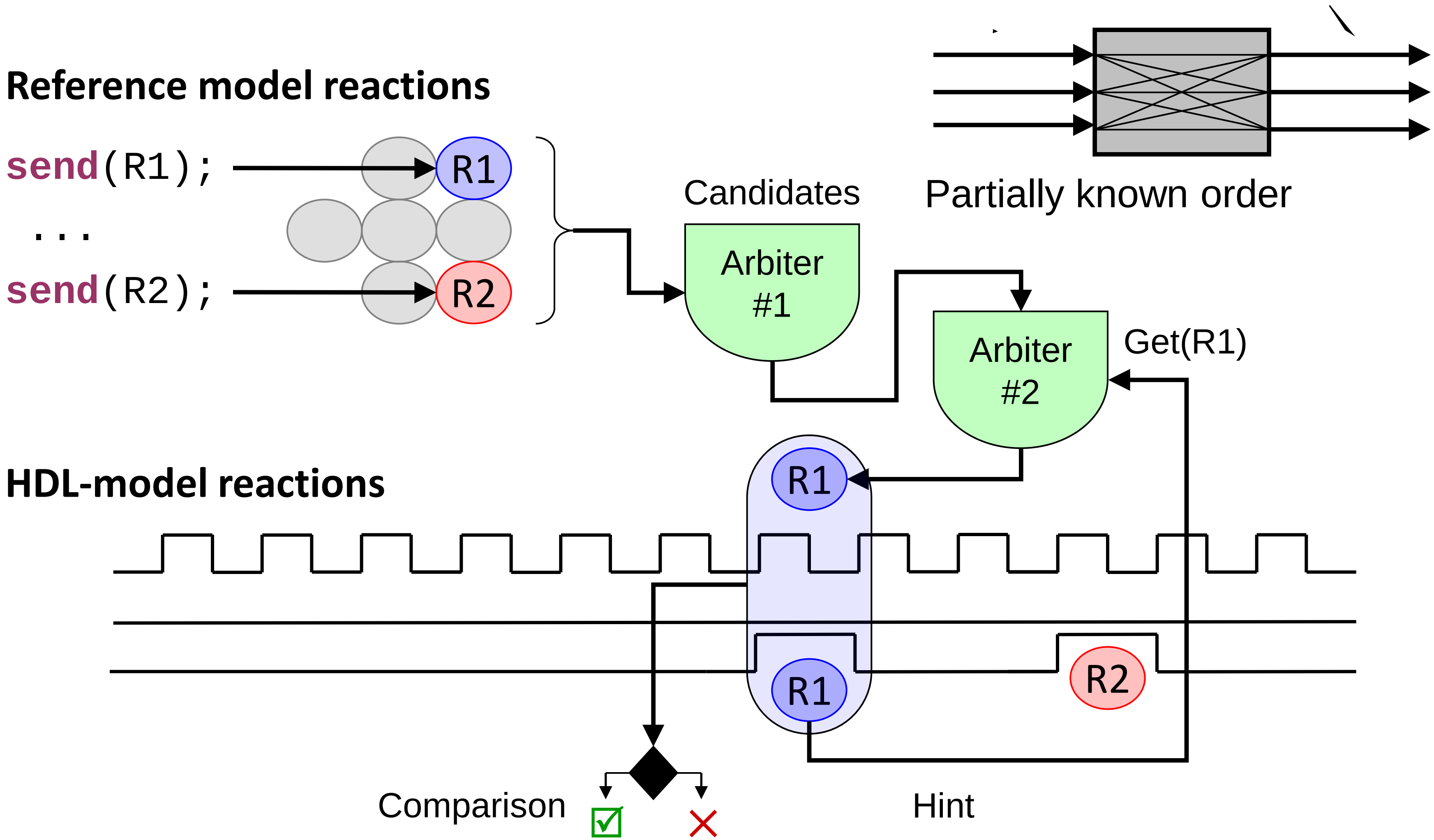
Deterministic arbiter



Adaptive arbiter



Two-level arbiter



Timed word (Alur & Dill, 1994)

$\mathbf{\Sigma}$ – alphabet of events
$\mathbf{T}$ – time domain ($\mathbf{R}^{\geq 0}$ or $\mathbf{N}$)

$w = (a_0, t_0)(a_1, t_1), \ldots \in (\mathbf{\Sigma} \times \mathbf{T})^{\omega(*)}$

- $\forall i \,.\, t_{\mathrm{i}} < t_{\mathrm{i+1}}$ ($t_{\mathrm{i}} \leq t_{\mathrm{i+1}}$) – monotonicity
- $\forall T \, \exists i \,.\, t_{\mathrm{i}} > T$ – progress (if $|w| = \infty$)

Mazurkiewicz trace (1977)

$\mathbf{\Sigma}$ – alphabet of events

$\mathbf{I} \subset \mathbf{\Sigma} \times \mathbf{\Sigma}$ – relation of independence

Equivalent: $u \equiv v \Leftrightarrow u$ is derived from v by means of reordering of closest independence events

Trace is a class of equivalence of event chains in respect to equivalent relation $\equiv$

Mazurkiewicz trace (1977) - Example

$\mathbf{\Sigma}$ = { a,b,c,d }
$\mathbf{I}$ = { (a,b), (c,d) + symmetry}

$[ab]_\equiv$ = { ab, ba }

$[bc]_\equiv$ = { bc }

$[abcd]_\equiv$ = { abcd, bacd, abdc, badc }

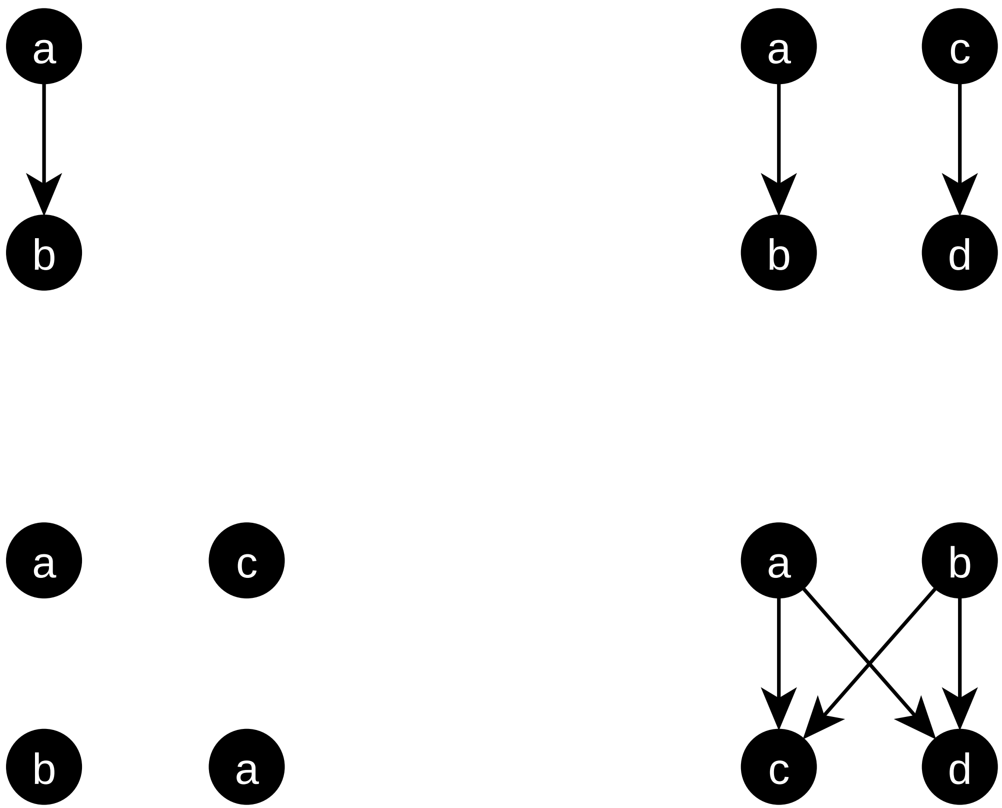
Partially ordered set – Pratt (1982)

$\boldsymbol{\Sigma}$ – alphabet of events

Pomset is tuple $\langle \mathbf{V}, \leq, \lambda \rangle$

- $\mathbf{V}$ – set of vertexes
- $\leq \subset \mathbf{V} \times \mathbf{V}$ – partial set
- $\lambda \colon \mathbf{V} \rightarrow \boldsymbol{\Sigma}$ – labeling function

Partially ordered set – Pratt (1982)
Examples



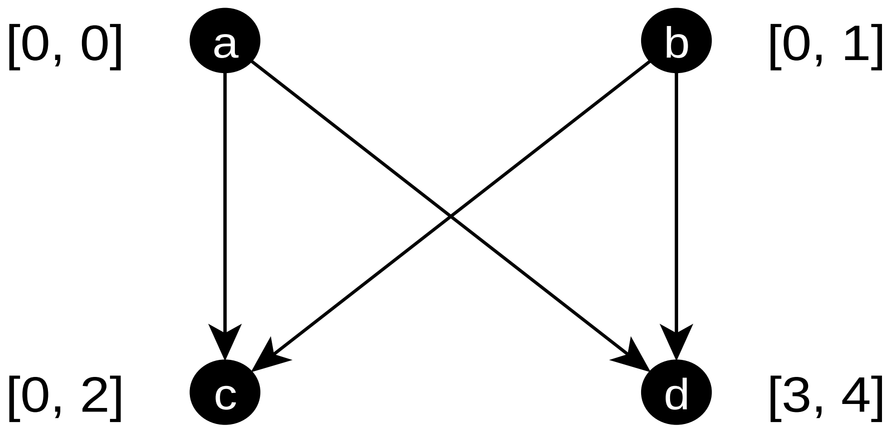
Timed trace – Chieu & Hung (2012)

$\mathbf{\Sigma}$ – alphabet of events, $\mathbf{T}$ – time domain

Timed trace – $\langle \mathbf{V}, \leq, \lambda, \theta\ [, \delta] \rangle$

- $\mathbf{V}$ – set of vertexes
- $\leq \subset \mathbf{V} \times \mathbf{V}$ – partial order
- λ: $\mathbf{V} \rightarrow \mathbf{\Sigma}$ – labeling function
- θ: $\mathbf{V} \rightarrow \mathbf{T}$ – time of event
- δ: $\mathbf{V} \rightarrow \Delta\mathbf{T}$ – allowed interval

Timed trace – Chieu & Hung (2012) Examples



- { abcd, bacd, abdc, badc }
- { abcd, bacd } – time restrictions

Behavior of specification and implementation

Implementation behavior

$\langle \mathbf{V}_I, \varnothing, \lambda_I, \theta_I \rangle$

Specification behavior

$\langle \mathbf{V}_S, \leq, \lambda_S, \theta_S, \delta_S \rangle$

Allowed time interval

$\delta_S(x) = [\theta_S(x) - \Delta\mathbf{t}(x), \theta_S(x) + \Delta\mathbf{t}(x)]$

Correspondence of events

$\mathbf{match}(\mathrm{x}, \mathrm{y}) = \big(\lambda_I(y) = \lambda_S(x)\big)\ \mathbf{\&}\ \big(\theta_I(y) \in \delta_S(x)\big)$

Conformance relation

$$I \sim S \Leftrightarrow \forall t \in \mathbf{T} \,.$$

$$\exists \mathbf{M} \subseteq \{\, (x, y) \in \mathbf{past}_S(t) \times \mathbf{past}_I(t) \mid \mathbf{match}(x, y) \,\}$$

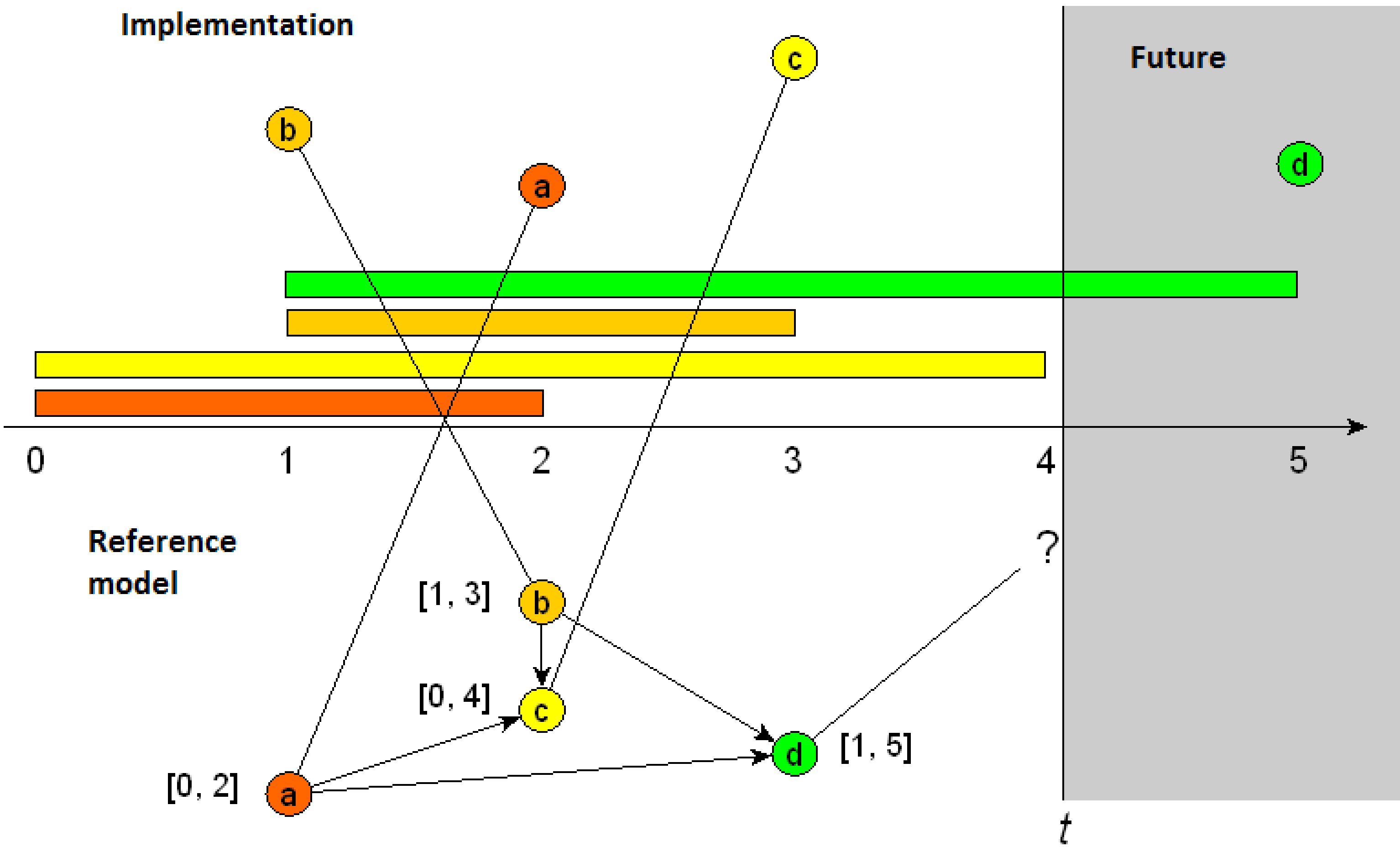
- $\mathbf{M}$ – one-to-one relation
- $\forall x \in \mathbf{past}_S(t\text{-}\mathbf{\Delta t})\ \exists y \in \mathbf{past}_I(t) \,.\, (x, y) \in \mathbf{M}$
- $\forall y \in \mathbf{past}_I(t\text{-}\mathbf{\Delta t})\ \exists x \in \mathbf{past}_S(t) \,.\, (x, y) \in \mathbf{M}$
- $\forall (x, y), (x', y') \in \mathbf{M} \,.\, x \leq x' \Rightarrow \boldsymbol{\theta}(y) \leq \boldsymbol{\theta}(y')$

Reaction arbiters

$$\mathbf{arbiter}_1 = \min_{\leq}(X)$$
$$X \subseteq \mathbf{V}_S$$

$$\mathbf{arbiter}_2(y, X) = \begin{cases} x, \exists x \in X \,.\, \mathbf{match}(x, y) \\ \epsilon, \nexists x \in X \,.\, \mathbf{match}(x, y) \end{cases}$$
$$y \in \mathbf{V}_I, X \subseteq \mathbf{V}_S$$

Conformance relation checking



C++TESK Testing ToolKit

Web: http://forge.ispras.ru/projects/cpptesk-toolkit

E-mail: cpptesk-support@ispras.ru

Home My page Projects Help

ISP RAS C++TESK Testing ToolKit

Overview Activity Roadmap Issues New issue Gantt Calendar News Documents Wiki Files Repository Hudson Settings

Files

File	Date	Size	D/L
1.0			
cpptesk-toolkit-1.0.1-beta-110415.tar.gz	04/15/2011 04:17 pm	2.1 MB	12
cpptesk-toolkit-1.0.2-beta-110504.tar.gz	05/04/2011 03:14 pm	2.6 MB	6
cpptesk-toolkit-1.0.3-beta-110510.tar.gz	05/10/2011 10:32 pm	4 MB	8
cpptesk-toolkit-1.0.4-beta-110520.tar.gz	05/20/2011 07:31 pm	5.9 MB	5
cpptesk-toolkit-1.0.5-beta-110528.tar.gz	05/28/2011 07:22 pm	6.5 MB	3
cpptesk-toolkit-1.0.6-beta-110621.tar.gz	06/21/2011 09:10 pm	6.8 MB	3
cpptesk-toolkit-1.0.7-beta-110625.tar.gz	06/25/2011 07:10 pm	7.5 MB	1
cpptesk-toolkit-src-1.0.1-beta-110415.tar.gz	04/15/2011 04:17 pm	916.3 kB	23
cpptesk-toolkit-src-1.0.2-beta-110504.tar.gz	05/04/2011 03:14 pm	3.8 MB	12
cpptesk-toolkit-src-1.0.3-beta-110510.tar.gz	05/10/2011 10:32 pm	5.4 MB	19
cpptesk-toolkit-src-1.0.4-beta-110520.tar.gz	05/20/2011 07:31 pm	7.6 MB	9
cpptesk-toolkit-src-1.0.5-beta-110528.tar.gz	05/28/2011 07:22 pm	8.7 MB	13
cpptesk-toolkit-src-1.0.6-beta-110621.tar.gz	06/21/2011 09:10 pm	9.2 MB	4
cpptesk-toolkit-src-1.0.7-beta-110625.tar.gz	06/25/2011 07:10 pm	10 MB	7

Conclusion

- Based on the theory of traces and partially ordered multisets method of on-the-fly analysis of hardware systems has been developed
- The method has been implemented in C++TESK Testing ToolKit and has been successfully used in a number of projects
- Future research is connected with failure diagnostics: giving hints to localization of bugs

THANK YOU

- Any questions?