

KSE 2014

October 10th, 2014

Hanoi

Modelling Timed Concurrent Systems Using Activity Diagram Patterns

Étienne André, Christine Choppy, Thierry Noulamo*

Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, France

*University of Dschang, IUT Fotso Victor, LAIA, Bandjoun, Cameroun

Introduction

- Activity diagrams: rich, permissive syntax, favours mistakes
- Semantics in natural language prevents formal verification

Introduction

- Activity diagrams: rich, permissive syntax, favours mistakes
- Semantics in natural language prevents formal verification

Our contributions

- Activity diagram patterns:
TADC (Timed Activity Diagram Components)
- Modular composition mechanism (also refinement)
- Semantics with time Petri nets

Outline

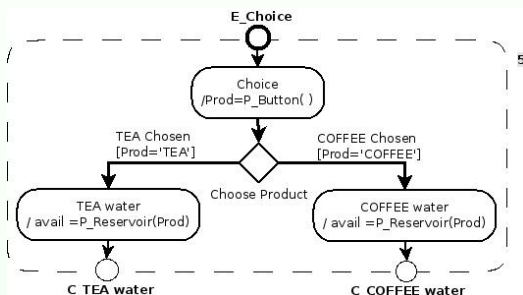
- 1 Timed Activity Diagrams Patterns
- 2 Translation into Time Petri Nets
- 3 Conclusion and Perspectives

Outline

- 1 Timed Activity Diagrams Patterns
- 2 Translation into Time Petri Nets
- 3 Conclusion and Perspectives

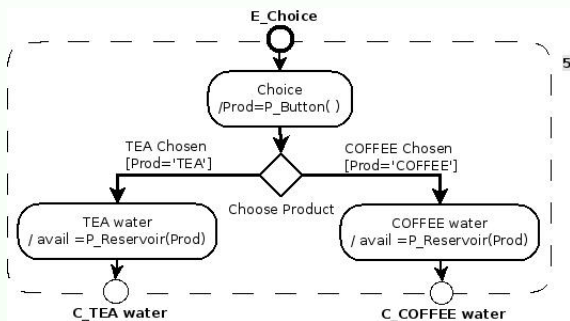
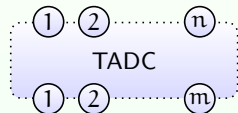
UML Activity Diagrams

- initial node, activity/flow final nodes, decision nodes (guards), fork/join nodes
- Global variables typed with **finite domains** (e.g. enumerated types)
- Activities may involve global variables **discrete, instantaneous** modifications (assignment, function call with side-effects)



Timed Activity Diagram Components (TADC)

- input connectors, output connectors
- “well-formed” activity diagrams, restricted construct set, adding timed constructs
- modular specification with possible refinement
- inductive mechanism

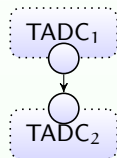


10 TADC patterns

- initial node, flow final node, simple activity ...

10 TADC patterns

- initial node, flow final node, simple activity ...
- sequence, non-deterministic delay, deterministic delay, deadline,

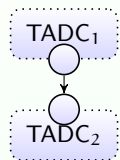


sequence

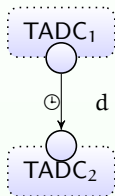
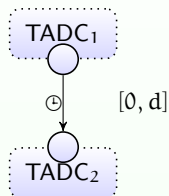
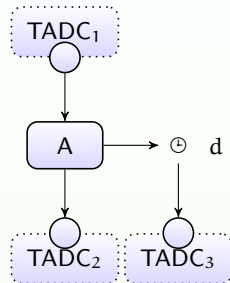
- input connectors = TADC₁ input connectors
- output connectors = TADC₂ output connectors

10 TADC patterns

- initial node, flow final node, simple activity ...
- sequence, non-deterministic delay, deterministic delay, deadline,



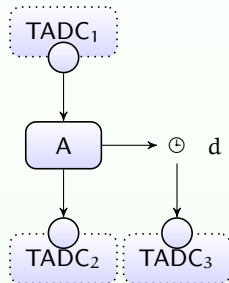
sequence

deterministic
delay $d \in \mathbb{R}_{\geq 0}$ non-determi-
-nistic delay

deadline

10 TADC patterns

- initial node, flow final node, simple activity ...
- sequence, non-deterministic delay, deterministic delay, deadline,

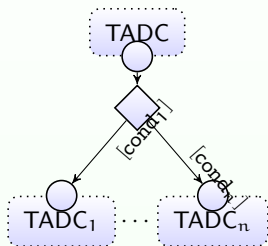


deadline

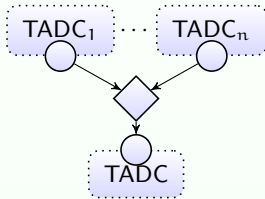
- TADC₁ execution
- activity A
- TADC₂ starts after at most d units of time
- alternatively, TADC₃ starts after exactly d units of time

10 TADC patterns (followed)

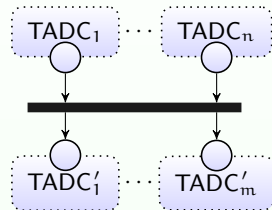
- decision, merge, synchronisation (cf UML)



decision



merge



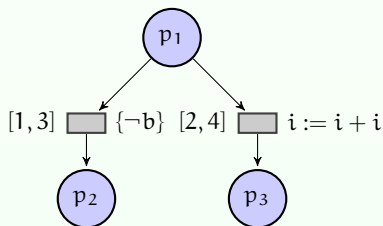
synchronisation

Outline

- 1 Timed Activity Diagrams Patterns
- 2 Translation into Time Petri Nets**
- 3 Conclusion and Perspectives

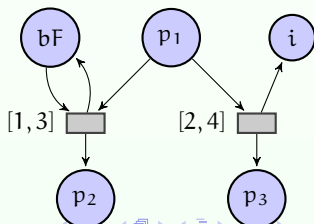
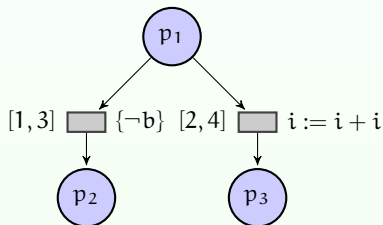
Time Petri Nets [Merlin, 1974] with global variables

- A kind of automaton
 - Bipartite graph with **places** (“standard” tokens) and **transitions**
 - Transitions associated with a **firing interval**
 - Enabled (tokens present) transitions must fire at the end of their firing interval (unless meanwhile disabled by another transition)
 - **Global variables** typed by a finite domain, used to express guards, updated when transition fired
 - Global variables: syntactic extension only
 - Note: Time Petri Nets with global variable: similarities with coloured Petri nets



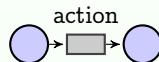
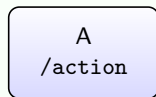
Time Petri Nets [Merlin, 1974] with global variables

- A kind of automaton
 - Bipartite graph with **places** (“standard” tokens) and **transitions**
 - Transitions associated with a **firing interval**
 - Enabled (tokens present) transitions must fire at the end of their firing interval (unless meanwhile disabled by another transition)
 - **Global variables** typed by a finite domain, used to express guards, updated when transition fired
 - Global variables: syntactic extension only
 - Note: Time Petri Nets with global variable: similarities with coloured Petri nets



Translation mechanism

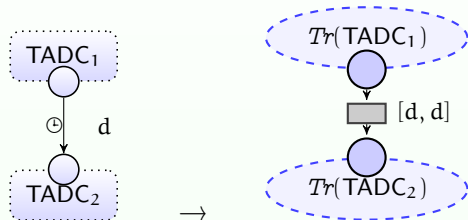
- each TADC is translated into a TPN fragment where the connectors are translated into places
- two TPN fragments can be composed by fusing the corresponding connector places together
- **simple activity**: TPN transition connected to input and output places (to be used for composition)



- assignments are easily translated
- functions involving a user input → non-deterministic choice

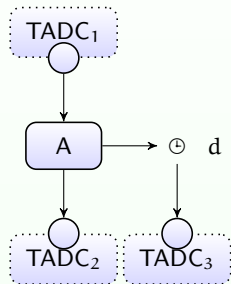
Translation mechanism (cont'd)

- Deterministic delay

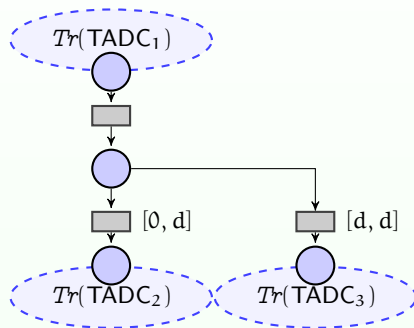


Translation mechanism (cont'd)

- Deadline

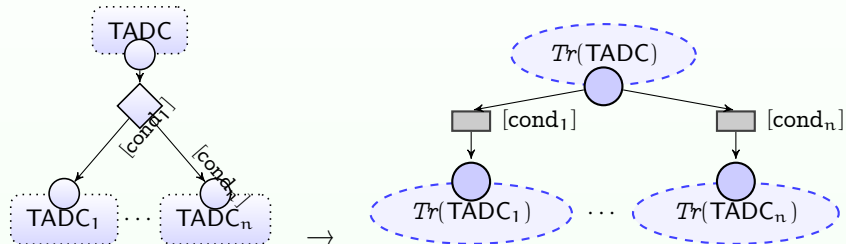


→



Translation mechanism (cont'd)

Decision



Outline

- 1 Timed Activity Diagrams Patterns
- 2 Translation into Time Petri Nets
- 3 Conclusion and Perspectives**

Conclusion

- Activity diagram patterns with **timed** constructs for **modular** composition
- **Semantics** in terms of time Petri nets
- [André et al., 2013]: “precise” activity diagram patterns to model business processes, modular, coloured Petri nets semantics
- Here: focus on time extension, less restrictive patterns (arbitrary number of input and output connectors), easier to “plug in” a higher level scheme.

Conclusion

Element	[UML, 2013]	[André et al., 2013]	This work
Activities	Yes	Yes	Yes
Data	Limited	Yes	Limited
Participants	Limited	Yes	No
Initial / final nodes	Yes	Yes	Yes
Decision	Yes	Restricted	Yes
Merge	Yes	Restricted	Yes
Fork	Yes	Restricted	Yes
Join	Yes	Restricted	Yes
Timed transitions	Limited	No	Yes

Table: Summary of the syntactic aspects considered

Perspectives

- Enrich with more complex features, e.g., timed synchronization of activities
- Refinement
- Tool ...

Bibliography

References I



(2013).

OMG unified modeling language. version 2.5 beta 2, 2013-09-05.
<http://www.omg.org/spec/UML/2.5/Beta2/PDF/>.



André, É., Choppy, C., and Reggio, G. (2013).

Activity diagrams patterns for modeling business processes.
In *SERA*, volume 496 of *Studies in Computational Intelligence*, pages 197–213.
Springer.



Jensen, K. and Kristensen, L. M. (2009).

Coloured Petri Nets – Modelling and Validation of Concurrent Systems.
Springer.



Merlin, P. M. (1974).

A study of the recoverability of computing systems.
PhD thesis, University of California, Irvine, CA, USA.