The Logic of Uppaal

2) \( = \).

\( (4 \xi \cdot \) \( \cdot \)

formulæ

\( F \) : formulæ

(27) \( \cdot \)

\( (16 \xi \cdot \)

1)

(0 \xi \cdot \)

2001)

\( (4 \xi \cdot \)

1999)

\( \cdot \)

\( (0) = \xi \cdot \)

Can it be empty?

Why is it a set?

• observer construction

• ad-hoc

• simple and wrong

Established DC Properties

• evolution

• satisfaction relation

• transition graph

Excursion:

• satisfaction relation

• observables

• between timed automata

Real-Time Systems

DC Properties for Timed Automata

Lecture 16: Automatic Verification of Real-Time Systems

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
Satisfaction of Uppaal Logic by Configurations

Example: Computation/Transition Graph

...
Example

Example
The text seems to be a continuation of a mathematical or technical discussion, possibly involving duration calculus and related concepts. The notation and symbols suggest a formal or theoretical context, which may be related to computer science or mathematics.

Uppaal Architecture

• Duration Calculus
• Constraint Diagrams
• PLC-Automata
• Networks of Timed Automata
• Region/Zone-Abstraction
• TA model-checking
• Extended Timed Automata
• Undecidability Results

Observables and Evolutions
• Duration Calculus (DC)
• Observables

Introduction

• DC Decidability
• DC Implementables
• PLC-Automata
• Timed Automata (TA), Uppaal
• Networks of Timed Automata
• Region/Zone-Abstraction
• TA model-checking
• Extended Timed Automata
• Undecidability Results

Recent Results:
• Timed Sequence Diagrams
• Quasi-equal Clocks
• Automatic Code Generation

Uppaal Query Language

• Syntax
• Syntax:
• Syntax:

Excursion: Transition Graph
• Satisfaction Relation
• Satisfaction relation between timed automata and DC formulae
• Observables of timed automata
• Evolution induced by computation path

Simple and Wrong Solution:
• Ad-hoc fix for invariants

Establishing DC Properties
• Observer Construction
• Untestable DC properties

Automatic Verification

...whether a TA satisfies a DC formula, observer-based...

...whether a TA satisfies a DC formula, observer-based...
In the following, we shall discuss the computation path \( 0_i = \varnothing \) \( \iff \) of network \( N \) \( \rightarrow \) \( N \) and set \( I_{N} \) of computation paths “Transform” each \( 0 \). . . .

The bright light off \( \varnothing \), \( \varnothing , \varnothing \), . . . .

Let \( \varnothing \) be a network of extended timed automata.

Observables of a Network of Timed Automata

Model Checking DC Properties with Uppaal

Characterise the behaviour of Two main options

For simplicity

Maybe: what kinds of DC formulae can we check with Uppaal?

Observing Automata Reference DC Properties

\( \varnothing \) \( \Rightarrow \) \( \varnothing \) \( \Rightarrow \) \( \varnothing \) \( \Rightarrow \) \( \varnothing \)

Example
Properties to be checked for a timed automata model can be specified using the Uppaal Query Language, which is a tiny little fragment of Timed CTL (TCTL), and as such by far not as expressive as Duration Calculus.

TCTL is another means to formalise requirements. For testable DC formulae $F$, we can automatically verify whether a network $N$ satisfies $F$ by constructing an observer automaton and transforming $N$ appropriately.

There are untestable DC formulae. (Everything else would be surprising.)

References

