Real-Time Systems

Lecture 7: DC Properties II

2017-11-16

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Content

- **RDC $+\ell = x, \forall\, x$ in Continous Time**
  - **Outline** of the proof
  - Recall: **two-counter** machines (2-CM)
    - **states** and **commands** (**syntax**)
    - **configurations** and **computations** (**semantics**)
  - Encoding **configurations** in DC
    - **initial configuration** of a 2-CM
  - Encoding **transitions** in DC
    - **increment** counter,
    - **decrement** counter,
    - and some helper formulae.
  - **Satisfiability and Validity**
  - **Discussion**

Decidability Results for Realisability: Overview

Fragment	Discrete Time	Continous Time
RDC	decidable ✓	decidable
RDC $+ \ell = r$	decidable for $r \in \mathbb{N}$	undecidable for $r \in \mathbb{R}^+$
RDC $+ \int P_1 = \int P_2$	undecidable	undecidable
RDC $+ \ell = x, \forall x$	undecidable	undecidable
DC	— " —	— " —

Decidability Results for RDC in Continuous Time

Recall: Restricted DC (RDC)

$$F ::= \lceil P \rceil \mid \neg F_1 \mid F_1 \vee F_2 \mid F_1 \,;\, F_2$$

where P is a state assertion with **boolean** observables **only**.

From now on: "RDC $+ \ell = x, \forall\, x$"

$$F ::= \lceil P \rceil \mid \neg F_1 \mid F_1 \vee F_2 \mid F_1 \,;\, F_2 \mid \ell = 1 \mid \ell = x \mid \forall\, x \bullet F_1$$

Undecidability of Satisfiability/Realisability from 0

Theorem 3.10.
The realisability from 0 problem for DC with **continuous time** is undecidable, not even semi-decidable.

Theorem 3.11.
The satisfiability problem for DC with continuous time is undecidable.

Sketch: Proof of Theorem 3.10

Reduce divergence of **two-counter machines** to realisability from 0:

- Given a two-counter machine $\mathcal{M}$ with final state q_{fin},
- construct a DC formula $F(\mathcal{M}) := encoding(\mathcal{M})$
- such that

$\mathcal{M}$ **diverges** **if and only if** the DC formula

$$F(\mathcal{M}) \land \neg\Diamond\lceil q_{fin}\rceil$$

is **realisable from 0**.

- If realisability from 0 was (semi-)decidable,
  divergence of two-counter machines would be (which it isn't).

Two-Counter Machines

Recall: Two-counter machines

A **two-counter** machine is a structure

$$\mathcal{M} = (\mathcal{Q}, q_0, q_{fin}, Prog)$$

where

- $\mathcal{Q}$ is a finite set of **states**,
- comprising the **initial state** q_0 and the **final state** q_{fin}
- $Prog$ is the **machine program**, i.e. a finite set of **commands** of the form

$$\underbrace{q : inc_i : q'} \quad \text{and} \quad \underbrace{q : dec_i : q', q'',} \qquad i \in \{1, 2\}.$$

q: $x_1 := x_1 + 1$; goto q'
q: $x_2 := x_2 + 1$; goto q'

q: if $(x_1 = 0)$ goto q'
else $x_1 := x_1 - 1$; goto q''

- We assume **deterministic** 2CM: for each $q \in \mathcal{Q}$, at most one command starts in q, and q_{fin} is the only state where no command starts.

2CM Configurations and Computations

- a **configuration** of $\mathcal{M}$ is a triple $K = (q, n_1, n_2) \in \mathcal{Q} \times \mathbb{N}_0 \times \mathbb{N}_0$.
- The **transition relation** "$\vdash$" on configurations is defined as follows:

Command	Semantics: $K \vdash K'$
$q : inc_1 : q'$	$(q, n_1, n_2) \vdash (q', n_1 + 1, n_2)$
$q : dec_1 : q', q''$	$(q, 0, n_2) \vdash (q', 0, n_2)$
	$(q, n_1 + 1, n_2) \vdash (q'', n_1, n_2)$
$q : inc_2 : q'$	$(q, n_1, n_2) \vdash (q', n_1, n_2 + 1)$
$q : dec_2 : q', q''$	$(q, n_1, 0) \vdash (q', n_1, 0)$
	$(q, n_1, n_2 + 1) \vdash (q'', n_1, n_2)$

- The (!) **computation** of $\mathcal{M}$ is a finite sequence of the form ("$\mathcal{M}$ **halts**")

$$K_0 = (q_0, 0, 0) \vdash K_1 \vdash K_2 \vdash \cdots \vdash (q_{fin}, n_1, n_2)$$

or an infinite sequence of the form ("$\mathcal{M}$ **diverges**")

$$K_0 = (q_0, 0, 0) \vdash K_1 \vdash K_2 \vdash \ldots$$

2CM Example

- $\mathcal{M} = (\mathcal{Q}, q_0, q_{fin}, Prog)$
- commands of the form $q : inc_i : q'$ and $q : dec_i : q', q''$, $i \in \{1, 2\}$
- configuration $K = (q, n_1, n_2) \in \mathcal{Q} \times \mathbb{N}_0 \times \mathbb{N}_0$.
-

Command	Semantics: $K \vdash K'$
$q : inc_1 : q'$	$(q, n_1, n_2) \vdash (q', n_1 + 1, n_2)$
$q : dec_1 : q', q''$	$(q, 0, n_2) \vdash (q', 0, n_2)$
	$(q, n_1 + 1, n_2) \vdash (q'', n_1, n_2)$
$q : inc_2 : q'$	$(q, n_1, n_2) \vdash (q', n_1, n_2 + 1)$
$q : dec_2 : q', q''$	$(q, n_1, 0) \vdash (q', n_1, 0)$
	$(q, n_1, n_2 + 1) \vdash (q'', n_1, n_2)$

$\mathcal{M}_1$

- $\mathcal{Q} = \{q_0, q_1, q_{fin}\}$
- $Prog = \{\underbrace{q_0 : inc_1 : q_1}_{1}, \underbrace{q_1 : inc_1 : q_{fin}}_{2}\}$

$(q_0, 0, 0)$
$\vdash$ 1
$(q_1, 1, 0)$
$\vdash$ 2
$(q_{fin}, 2, 0)$

$\hookrightarrow$ $\mathcal{M}_1$ halts

$\mathcal{M}_2$

- $\mathcal{Q} = \{q_0, q_{fin}\}$
- $Prog = \{\underbrace{q_0 : inc_2 : q_0}_{1}\}$

$(q_0, 0, 0)$
$\vdash$ 1
$(q_0, 0, 1)$
$\vdash$ 1
$(q_0, 0, 2)$
$\vdash$
$\vdots$

$\hookrightarrow$ $\mathcal{M}_2$ diverges

Reduction to 2-CM: Idea

Reducing Divergence to DC realisability: Idea In Pictures

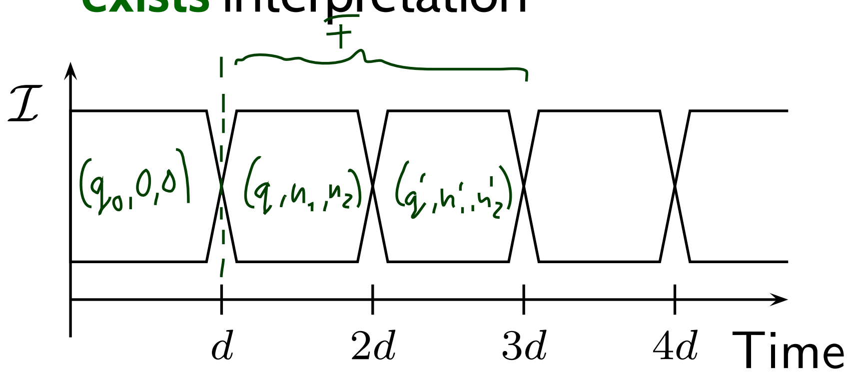
2CM $\mathcal{M}$ **diverges**

iff

exists $\pi : K_0 \vdash K_1 \vdash \dots$

iff

exists interpretation



"$\mathcal{I}$ describes π"
and
$\mathcal{I} \models_0 F(\mathcal{M}) \land \neg\Diamond\lceil q_{fin} \rceil$

$F(\mathcal{M})$ intuitively specifies:

- $[0, d]$ encodes $(q_0, 0, 0)$,
- each $[n \cdot d, (n+1) \cdot d]$ encodes a configuration,
- $[n \cdot d, (n+1) \cdot d]$ and $[(n+1) \cdot d, (n+2) \cdot d]$ are in $\vdash$-relation,
- if q_{fin} is reached, we stay there

Reducing Divergence to DC realisability: Idea

"(q, n_1, n_2)"

- A single configuration K of $\mathcal{M}$ can be encoded in an interval of length 4; **being an encoding interval** can be **characterised** by a DC formula.

- An interpretation on 'Time' encodes **the** computation of $\mathcal{M}$ if
  - each interval $[4n, 4(n+1)]$, $n \in \mathbb{N}_0$, **encodes** a configuration K_n,
  - each two subsequent intervals

    $$[4n, 4(n+1)] \text{ and } [4(n+1), 4(n+2)], n \in \mathbb{N}_0,$$

    encode configurations $K_n \vdash K_{n+1}$ **in transition relation**.

- **Being an encoding of the run** can be **characterised** by a DC formula $F(\mathcal{M})$.

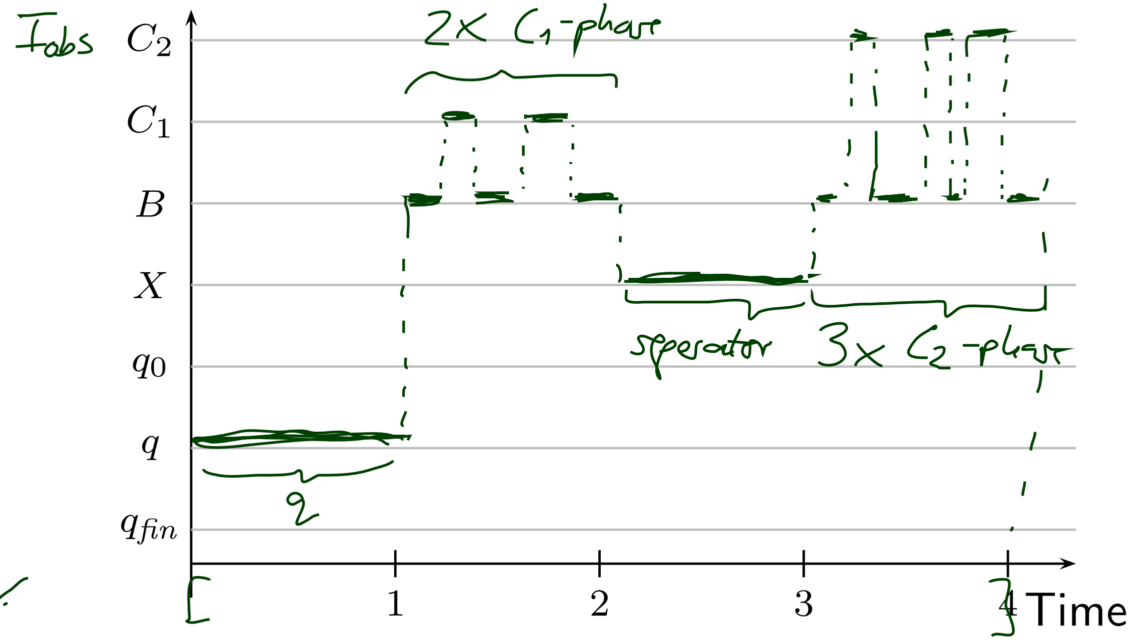
- Then $\mathcal{M}$ **diverges** if and only if $F(\mathcal{M}) \wedge \neg\Diamond\lceil q_{fin} \rceil$ is realisable from 0.

Encoding Configurations

Encoding Configurations

- We use Obs $= \{\text{obs}\}$ with
  $\mathcal{D}(\text{obs}) = \mathcal{Q}_{\mathcal{M}} \dot\cup \{C_1, C_2, B, X\}$.

  disjoint



Examples:

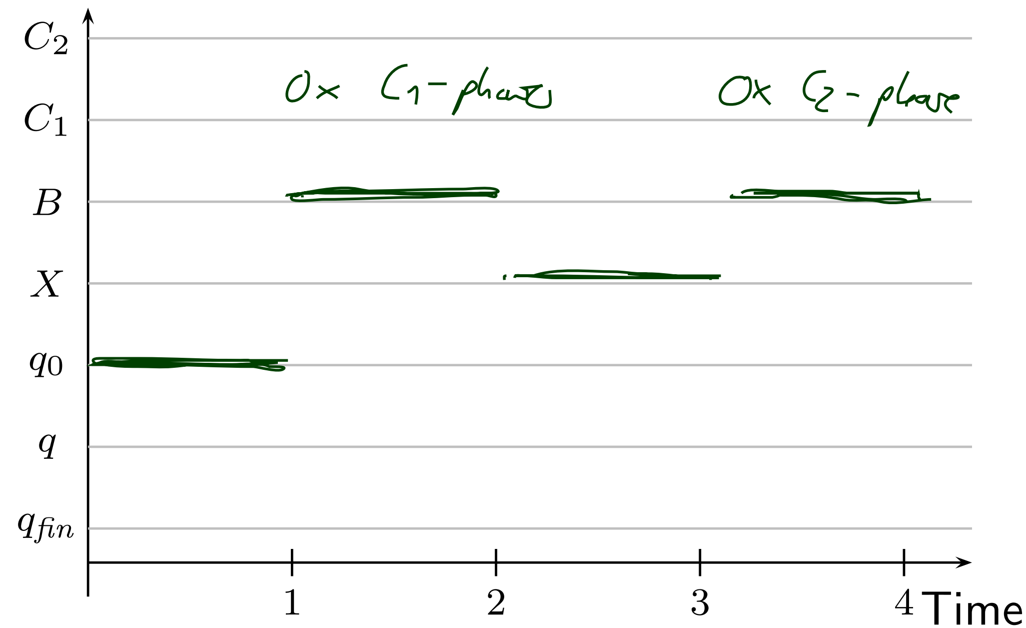
- $K = (q, 2, 3)$

$$\left(\begin{array}{c}\lceil q \rceil \\ \wedge \\ \ell = 1\end{array}\right) ; \left(\begin{array}{c}\lceil B \rceil ; \lceil C_1 \rceil ; \lceil B \rceil ; \lceil C_1 \rceil ; \lceil B \rceil \\ \wedge \\ \ell = 1\end{array}\right) ; \left(\begin{array}{c}\lceil X \rceil \\ \wedge \\ \ell = 1\end{array}\right) ; \left(\begin{array}{c}\lceil B \rceil ; \lceil C_2 \rceil ; \lceil B \rceil ; \lceil C_2 \rceil ; \lceil B \rceil ; \lceil C_2 \rceil ; \lceil B \rceil \\ \wedge \\ \ell = 1\end{array}\right)$$

Encoding Configurations

- We use Obs $= \{\text{obs}\}$ with
  $\mathcal{D}(\text{obs}) = \mathcal{Q}_\mathcal{M} \,\dot{\cup}\, \{C_1, C_2, B, X\}$.

  disjoint



Examples:

- $K = (q, 2, 3)$

$$\begin{pmatrix} \lceil q \rceil \\ \wedge \\ \ell = 1 \end{pmatrix} ; \begin{pmatrix} \lceil B \rceil ; \lceil C_1 \rceil ; \lceil B \rceil ; \lceil C_1 \rceil ; \lceil B \rceil \\ \wedge \\ \ell = 1 \end{pmatrix} ; \begin{pmatrix} \lceil X \rceil \\ \wedge \\ \ell = 1 \end{pmatrix} ; \begin{pmatrix} \lceil B \rceil ; \lceil C_2 \rceil ; \lceil B \rceil ; \lceil C_2 \rceil ; \lceil B \rceil ; \lceil C_2 \rceil ; \lceil B \rceil \\ \wedge \\ \ell = 1 \end{pmatrix}$$

- $K_0 = (q_0, 0, 0)$

$$\begin{pmatrix} \lceil q_0 \rceil \\ \wedge \\ \ell = 1 \end{pmatrix} ; \begin{pmatrix} \lceil B \rceil \\ \wedge \\ \ell = 1 \end{pmatrix} ; \begin{pmatrix} \lceil X \rceil \\ \wedge \\ \ell = 1 \end{pmatrix} ; \begin{pmatrix} \lceil B \rceil \\ \wedge \\ \ell = 1 \end{pmatrix}$$

or, using abbreviations, $\lceil q_0 \rceil^1 ; \lceil B \rceil^1 ; \lceil X \rceil^1 ; \lceil B \rceil^1$.

Formula Construction for Given 2-CM

Construction of $F(\mathcal{M})$

In the following, we give **DC formulae describing**

- the **initial configuration**: $init$,
- the **general form of configurations**: $keep$,
- the **transitions between configurations**: $F(q : inc_i : q')$ and $F(q : dec_i : q')$,
- the handling of the **final state**.

$F(\mathcal{M})$ is the conjunction of all these formulae:

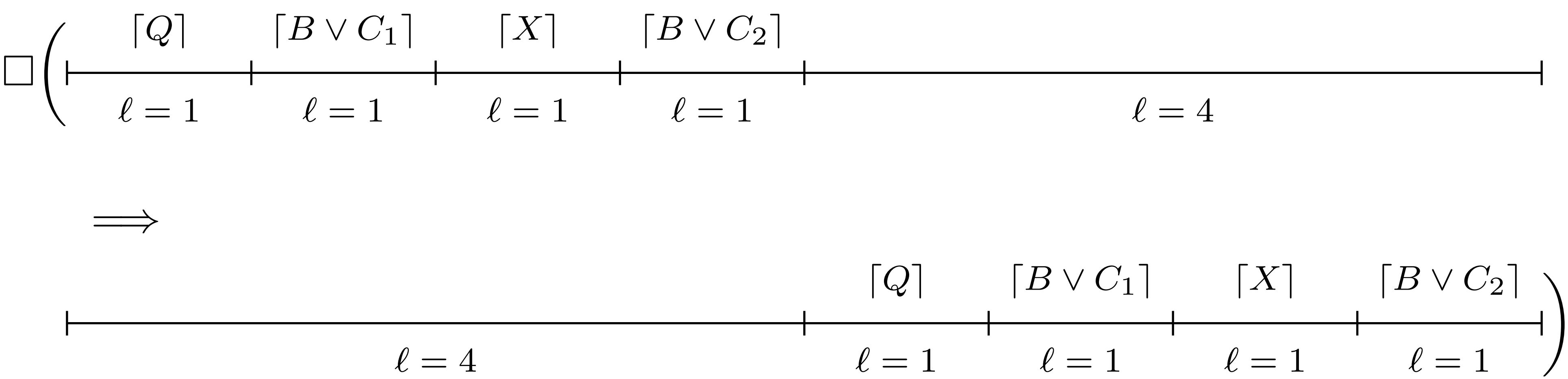
$$F(\mathcal{M}) = init \wedge keep \wedge \dots$$
$$\wedge \bigwedge_{q:inc_i:q' \in Prog} F(q : inc_i : q')$$
$$\wedge \bigwedge_{q:dec_i:q' \in Prog} F(q : dec_i : q')$$

Initial and General Configurations

$$init :\iff (\ell \geq 4 \implies \lceil q_0 \rceil^1 \, ; \lceil B \rceil^1 \, ; \lceil X \rceil^1 \, ; \lceil B \rceil^1 \, ; true)$$

$$keep :\iff \Box((\lceil Q \rceil^1 \, ; \lceil B \vee C_1 \rceil^1 \, ; \lceil X \rceil^1 \, ; \lceil B \vee C_2 \rceil^1 \, ; \ell = 4)$$
$$\implies (\ell = 4 \, ; \lceil Q \rceil^1 \, ; \lceil B \vee C_1 \rceil^1 \, ; \lceil X \rceil^1 \, ; \lceil B \vee C_2 \rceil^1))$$

where $Q := \neg(X \vee C_1 \vee C_2 \vee B)$.



Auxiliary Formula Pattern copy

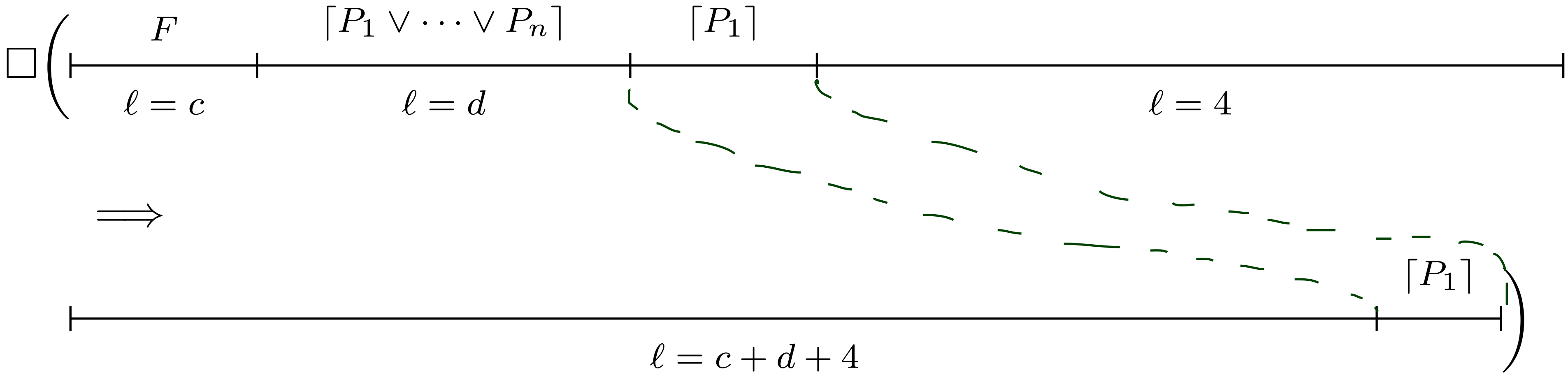
formula   state assertions

$$copy(F, \{P_1, \dots, P_n\}) :\iff$$

$$\forall\, c, d \bullet \Box((F \land \ell = c) \,;\, (\lceil P_1 \lor \dots \lor P_n \rceil \land \ell = d) \,;\, \lceil P_1 \rceil \,;\, \ell = 4$$
$$\implies \ell = c + d + 4 \,;\, \lceil P_1 \rceil)$$

$\land \dots$

$$\forall\, c, d \bullet \Box((F \land \ell = c) \,;\, (\lceil P_1 \lor \dots \lor P_n \rceil \land \ell = d) \,;\, \lceil P_n \rceil \,;\, \ell = 4$$
$$\implies \ell = c + d + 4 \,;\, \lceil P_n \rceil)$$

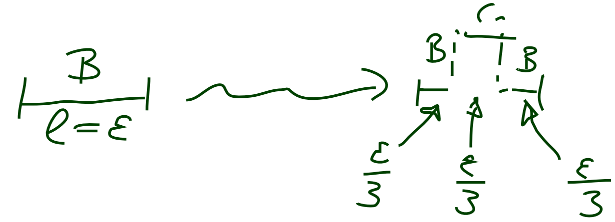


$q : inc_1 : q'$ (*Increment*)

(i) Change state

$$\Box(\lceil q \rceil^1 \,;\, \lceil B \vee C_1 \rceil^1 \,;\, \lceil X \rceil^1 \,;\, \lceil B \vee C_2 \rceil^1 \,;\, \ell = 4 \implies \ell = 4 \,;\, \lceil q' \rceil^1 \,;\, true)$$

$q : inc_1 : q'$ *(Increment)*



(i) Change state

$$\Box(\lceil q\rceil^1 ; \lceil B \vee C_1\rceil^1 ; \lceil X\rceil^1 ; \lceil B \vee C_2\rceil^1 ; \ell = 4 \implies \ell = 4 ; \lceil q'\rceil^1 ; true)$$

$\Box\Big($ $\lceil q\rceil$ ($\ell = 1$) | $\lceil B \vee C_1\rceil$ ($\ell = 1$) | $\lceil X\rceil$ ($\ell = 1$) | $\lceil B \vee C_2\rceil$ ($\ell = 1$) | ($\ell = 4$)

$\implies$ ($\ell = 4$) | $\lceil q'\rceil$ ($\ell = 1$) | $true$ $\Big)$

(ii) Increment counter

$$\forall\, d \bullet \Box(\lceil q\rceil^1 ; \lceil B\rceil^d ; (\ell = 0 \vee \lceil C_1\rceil ; \lceil \neg X\rceil) ; \lceil X\rceil^1 ; \lceil B \vee C_2\rceil^1 ; \ell = 4$$
$$\implies \ell = 4 ; \lceil q'\rceil^1 ; (\lceil B\rceil ; \lceil C_1\rceil ; \lceil B\rceil \wedge \ell = d) ; true$$

old counter value

$\forall\, d \bullet \Box\Big($ q ($\ell = 1$) | $\lceil B\rceil$ ($\ell = d$) | $\ell = 0 \vee (\lceil C_1\rceil ; \lceil \neg X\rceil)$ | $\lceil X\rceil$ ($\ell = 1$) | $\lceil B \vee C_2\rceil$ ($\ell = 1$) | ($\ell = 4$)

$\implies$ ($\ell = 4$) | $\lceil q'\rceil$ ($\ell = 1$) | $\lceil B\rceil ; \lceil C_1\rceil ; \lceil B\rceil$ ($\ell = d$) | $true$ $\Big)$

$q : inc_1 : q'$ *(Increment)*

(i) Keep rest of first counter

$$copy(\underbrace{\lceil q \rceil^1 \,;\, \lceil B \vee C_1 \rceil \,;\, \lceil C_1 \rceil}_{F}, \underbrace{\{B, C_1\}}_{\{P_1, P_2\}})$$

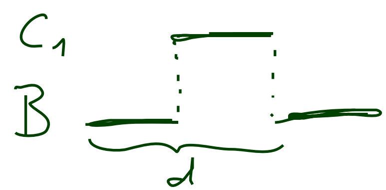
(ii) Leave second counter unchanged

$$copy(\lceil q \rceil^1 \,;\, \lceil B \vee C_1 \rceil \,;\, \lceil X \rceil^1, \{B, C_2\})$$

$q : dec_1 : q', q''$ *(Decrement)*

(i) If zero

$$\Box(\lceil q \rceil^1 ; \lceil B \rceil^1 ; \lceil X \rceil^1 ; \lceil B \vee C_2 \rceil^1 ; \ell = 4 \implies \ell = 4 ; \lceil q' \rceil^1 ; \lceil B \rceil^1 ; \mathit{true})$$



(ii) Decrement counter

$$\forall\, d \bullet \Box(\lceil q \rceil^1 ; (\lceil B \rceil ; \lceil C_1 \rceil \wedge \ell = d) ; \lceil B \rceil ; \lceil B \vee C_1 \rceil ; \lceil X \rceil^1 ; \lceil B \vee C_2 \rceil^1 ; \ell = 4$$
$$\implies \ell = 4 ; \lceil q'' \rceil^1 ; \lceil B \rceil^d ; \mathit{true})$$

(iii) Keep rest of first counter

$$\mathit{copy}(\lceil q \rceil^1 ; \lceil B \rceil ; \lceil C_1 \rceil ; \lceil B_1 \rceil, \{B, C_1\})$$

(iv) Leave second counter unchanged

$$\mathit{copy}(\lceil q \rceil^1 ; \lceil B \vee C_1 \rceil ; \lceil X \rceil^1, \{B, C_2\})$$

Final State

$$copy(\underbrace{\lceil q_{fin} \rceil^1 ; \lceil B \vee C_1 \rceil^1 ; \lceil X \rceil ; \lceil B \vee C_2 \rceil^1}_{F}, \underline{\{q_{fin}, B, X, C_1, C_2\}})$$

$\mathcal{M}$ diverges

iff

$F(\mathcal{M}) \wedge \neg \Diamond \lceil q_{fin} \rceil$

is realisable from 0

Satisfiability / Valididty

Satisfiability

- Following Chaochen and Hansen (2004) we can observe that

  $\mathcal{M}$ **halts if and only if** the DC formula $F(\mathcal{M}) \wedge \Diamond\lceil q_{fin} \rceil$ is **satisfiable**.

  This yields

  > **Theorem 3.11.**
  > The satisfiability problem for DC with continuous time is undecidable.

  (It is semi-decidable.)

- Furthermore, by taking the contraposition, we see

  $\mathcal{M}$ **diverges** **if and only if** $\mathcal{M}$ does not **halt**
  if and only if $F(\mathcal{M}) \wedge \neg\Diamond\lceil q_{fin} \rceil$ is **not** satisfiable.

- Thus whether a DC formula is **not satisfiable** is not decidable,
  not even semi-decidable.

Validity

- By Remark 2.13, F is valid iff $\neg F$ is not satisfiable, so

> **Corollary 3.12.** The validity problem for DC with continuous time is undecidable, not even semi-decidable.

- This provides us with an alternative proof of Theorem 2.23 ("there is no sound and complete proof system for DC"):
  - **Suppose** there were such a calculus $\mathcal{C}$.
  - By Lemma 2.22 it is semi-decidable whether a given DC formula F is a theorem in $\mathcal{C}$.
  - By the soundness and completeness of $\mathcal{C}$, F is a theorem in $\mathcal{C}$ **if and only if** F is valid.
  - Thus it is semi-decidable whether F is valid. **Contradiction.**

Discussion

- Note: the DC fragment defined by the following grammar is **sufficient** for the reduction

$$F ::= \lceil P \rceil \mid \neg F_1 \mid F_1 \vee F_2 \mid F_1 \,;\, F_2 \mid \ell = 1 \mid \ell = x \mid \forall\, x \bullet F_1,$$

  P a state assertion, x a global variable.

- Formulae used in the reduction are abbreviations:

$$\begin{aligned} \ell = 4 &\iff \ell = 1 \,;\, \ell = 1 \,;\, \ell = 1 \,;\, \ell = 1 \\ \ell \geq 4 &\iff \ell = 4 \,;\, \mathit{true} \\ \ell = x + y + 4 &\iff \ell = x \,;\, \ell = y \,;\, \ell = 4 \end{aligned}$$

- Length 1 is not necessary – we can use $\ell = z$ instead, with fresh z.

- This is RDC augmented by "$\ell = x$" and "$\forall\, x$",
  which we denote by **RDC** $+ \ell = x, \forall\, x$.

Content

- **RDC $+\ell = x, \forall\, x$ in Continous Time**
  - **Outline** of the proof
  - Recall: **two-counter** machines (2-CM)
    - **states** and **commands** (**syntax**)
    - **configurations** and **computations** (**semantics**)
  - Encoding **configurations** in DC
    - **initial configuration** of a 2-CM
  - Encoding **transitions** in DC
    - **increment** counter,
    - **decrement** counter,
    - and some helper formulae.
  - **Satisfiability and Validity**
  - **Discussion**

Tell Them What You've Told Them...

- For **Restricted DC** plus $\ell = x$ and $\forall\, x$ in continuous time:
  - **satisfiability** is **undecidable**.
  - **Proof idea**: reduce to halting problem of two-counter machines.
- For full DC, it doesn't get better.

Content

Introduction

- **Observables and Evolutions** ✓
- **Duration Calculus** (DC) ✓
- Semantical Correctness Proofs ✓
- DC Decidability ✓
- DC Implementables
- **PLC-Automata**

8–10

$obs : \mathsf{Time} \to \mathscr{D}(obs)$

- **Timed Automata** (TA), Uppaal
- Networks of Timed Automata
- Region/Zone-Abstraction
- TA model-checking
- Extended Timed Automata
- Undecidability Results

$\langle obs_0, \nu_0 \rangle, t_0 \xrightarrow{\lambda_0} \langle obs_1, \nu_1 \rangle, t_1 \ldots$

- **Automatic Verification**...

  ...whether a TA satisfies a DC formula, observer-based
- **Recent Results**:
  - **Timed Sequence Diagrams**, or **Quasi-equal Clocks**, or **Automatic Code Generation**, or ...

References

References

Chaochen, Z. and Hansen, M. R. (2004). *Duration Calculus: A Formal Approach to Real-Time Systems*. Monographs in Theoretical Computer Science. Springer-Verlag. An EATCS Series.

Olderog, E.-R. and Dierks, H. (2008). *Real-Time Systems - Formal Specification and Automatic Verification*. Cambridge University Press.