Real-Time Systems

Lecture 6: DC Properties I

2017-11-14

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Content

Introduction

- **Observables and Evolutions**
- **Duration Calculus** (DC)
- Semantical Correctness Proofs ✓
- 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

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

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

- **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 ...

Content

- **A Calculus for DC**: A brief outlook
  - Recall: **predicate calculus**
  - DC Calculus is **just the same**, just a few more rules
  - → cf. textbook Olderog/Dierks
- **Decidability Results for DC**: Motivation
- **RDC in Discrete Time**
  - Restricted DC **syntax**
  - **Discrete time interpretation** of RDC
  - **Discrete** vs. continuous time
  - The **satisfiability** problem for RDC / discrete time
    - The **language** of a formula

Recall: Predicate Calculus

Recall: Calculus

- A **proof system** or **calculus** $\mathcal{C}$ is a finite set of **proof rules** of the form

$$\frac{F_1, \ldots, F_n}{F} \quad \text{where} \quad cond(F_1, \ldots, F_n, F)$$

consisting of:

- **premise**, $n \geq 0$
- **conclusion**,
- **application condition** (has to be **decidable**).

- In case $n = 0$, the rule is called **axiom** and written as

$$F \quad \text{where} \quad cond(F)$$

- If the **application condition** is a **tautology**, we may **omit** it.

Recall: Proofs in a Calculus

The central concepts of a calculus are that of **proof** and **provability**.

- A **proof** of a formula F in $\mathcal{C}$ **from** a set of formulae $\mathcal{H}$ is a finite sequence

$$\begin{matrix} G_1 \\ \vdots \\ G_m \end{matrix}$$

such that each formula $G_i, 1 \leq i \leq m,$

- G_i is in $\mathcal{H}$ (called **assumption** or **hypothesis**), or
- G_i is an **axiom** of $\mathcal{C}$,
- G_i is a **conclusion of a rule** in $\mathcal{C}$ applied to some predecessor formulae in the proof, i.e. there exists a proof rule

$$(*) \quad \frac{F_1, \ldots, F_n}{G_i} \quad \text{where} \quad cond(F_1, \ldots, F_n, G_i)$$

s.t. $F_1, \ldots, F_n \subseteq \{G_1, \ldots, G_{i-1}\}$ and $cond(F_1, \ldots, F_n, G_i)$ holds.

Example: Predicate Calculus

- T: it is Tue or Thu between 14:00 and 16:00
- R: I'm in the RTS lecture
- E: I'm excited

Assumptions $\mathcal{H}$:

① $T \implies R$ (on Tue/Thu times, I'm at RTS lecture)
② $R \implies E$ (in the RTS lecture, I'm excited)
③ $\neg E$ (I'm not excited now)

Claim: $\mathcal{H} \models \neg T$ (If $\mathcal{H}$ hold, it's not Tue/Wed 14:00-16:00 now.)

Some predicate calculus proof rules:

(A) $\dfrac{p \implies q, \quad q \implies r}{p \implies r}$ (B) contraposition $\dfrac{p \implies q}{\neg q \implies \neg p}$ (C) modus ponens $\dfrac{p \implies q, \quad p}{q}$

① ②
apply (A)
$T \implies E$

Example: Predicate Calculus

- T: it is Tue or Thu between 14:00 and 16:00
- R: I'm in the RTS lecture
- E: I'm excited

Assumptions $\mathcal{H}$:

① $T \implies R$ (on Tue/Thu times, I'm at RTS lecture)
② $R \implies E$ (in the RTS lecture, I'm excited)
③ $\neg E$ (I'm not excited now)

Claim: $\mathcal{H} \models \neg T$ (If $\mathcal{H}$ hold, it's not Tue/Wed 14:00-16:00 now.)

Some predicate calculus proof rules:

(A) $\dfrac{p \implies q, \quad q \implies r}{p \implies r}$ (B) contraposition $\dfrac{p \implies q}{\neg q \implies \neg p}$ (C) modus ponens $\dfrac{p \implies q, \quad p}{q}$

$\dfrac{p}{\neg p}$

① ②
apply (A)
$T \implies E$
apply (B)
$\neg E \implies \neg T$ ③
apply (C)
$\neg T$

Thus $\mathcal{H} \vdash \neg T$.

Recall: Theorems of a Calculus

- We say, F is **provable** from $\mathcal{H} = \{H_1, \dots, H_k\}$ in $\mathcal{C}$, in symbols

$$\mathcal{H} \vdash_{\mathcal{C}} F,$$

if and only if there **exists a proof** of F from $\mathcal{H}$ in $\mathcal{C}$.

- **Notation:**
  - write $H_1, \dots, H_k \vdash_{\mathcal{C}} F$ instead of $\{H_1, \dots, H_k\} \vdash_{\mathcal{C}} F$;
  - write $\vdash_{\mathcal{C}} F$ instead of $\emptyset \vdash_{\mathcal{C}} F$;
  - If $\mathcal{C}$ is clear from the context, we may omit the index.

- A formula F with $\vdash_{\mathcal{C}} F$ is called a **theorem** of $\mathcal{C}$.

Recall: Soundness and Completeness of a Calculus

- A calculus $\mathcal{C}$ is called **sound** (or correct) if and only if

$$\mathcal{H} \vdash_{\mathcal{C}} F \text{ implies } \mathcal{H} \models F$$

"whenever F is **(syntactically) derivable** from $\mathcal{H}$ in $\mathcal{C}$, then F is **implied** by $\mathcal{H}$ **semantically**".

In case of DC, "$\mathcal{H} \models F$" means:

for all interpretations $\mathcal{I}$, if $\mathcal{I} \models G$ (realise) for all $G \in \mathcal{H}$ then $\mathcal{I} \models F$.

- To be useful, a calculus (for DC) should be sound.

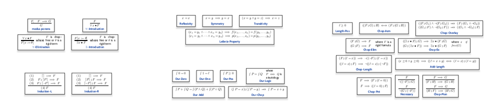
- A calculus $\mathcal{C}$ is called **complete** if and only if

$$\mathcal{H} \models F \text{ implies } \mathcal{H} \vdash_{\mathcal{C}} F$$

- Due to reasons of computability, we cannot always have completeness.

A Calculus for DC

A Sound Calculus for DC



A Sound Calculus for DC

$$\frac{F, \quad F \implies G}{G}$$
modus ponens

$$\frac{F}{\forall x \bullet F}$$
$\forall$-Introduction

$$\frac{\forall x \bullet F}{F[x := \theta]}$$ where F is chop-free or θ is a rigid term
$\forall$-Elimination

$$\frac{F[x := \theta]}{\exists x \bullet F}$$ where F is chop-free or θ is a rigid term
$\exists$-Introduction

Predicate Calculus

A Sound Calculus for DC

$x = x$
Reflexivity

$x = y \implies y = x$
Symmetry

$(x = y \wedge y = z) \implies x = z$
Transitivity

$$(x_1 = y_1 \wedge \dots \wedge x_n = y_n) \implies f(x_1, \dots, x_n) = f(y_1, \dots, y_n)$$
$$(x_1 = y_1 \wedge \dots \wedge x_n = y_n) \implies p(x_1, \dots, x_n) = p(y_1, \dots, y_n)$$
Leibniz-Property

Axiomatisation of Equality

A Sound Calculus for DC

$\ell \geq 0$
Length-Pos

$((F ; G) ; H) \iff (F ; (G ; H))$
Chop-Asm

$$((F ; G_1) \wedge \neg(F ; G_2)) \implies (F ; (G_1 \wedge \neg G_2))$$
$$((G_1 ; F) \wedge \neg(G_2 ; F)) \implies ((G_1 \wedge \neg G_2) ; F)$$
Chop-Overlay

$(F ; G) \implies F$
$(G ; F) \implies F$ where F is a rigid formula
Chop-Elim

$((\exists x \bullet F) ; G) \implies \exists x \bullet (F ; G)$
$(G ; (\exists x \bullet F)) \implies \exists x \bullet (G ; F)$ where $x \notin \mathit{free}(G)$
Chop-Ex

$(F ; (\ell = x)) \implies \neg((\neg F) ; (\ell = x))$
$((\ell = x) ; F) \implies \neg((\ell = x) ; (\neg F))$
Chop-Length

$(x \geq 0 \wedge y \geq 0) \implies ((\ell = x + y) \iff (\ell = x) ; (\ell = y))$
Add-Length

$F \implies (F ; (\ell = 0))$
$F \implies ((\ell = 0) ; F)$
Chop-Pnt

$$\frac{F}{\neg((\neg F) ; G)} \qquad \frac{F}{\neg(G ; (\neg F))}$$
Necessary

$$\frac{F \implies G}{(F ; H) \implies (G ; H)} \qquad \frac{F \implies G}{(H ; F) \implies (H ; G)}$$
Chop-Mon

Interval Logic

A Sound Calculus for DC

$\int 0 = 0$
Dur-Zero

$\int 1 = \ell$
Dur-One

$\int P \geq 0$
Dur-Pos

$\int P = \int Q$ where $P \iff Q$ is a tautology
Dur-Logic

$\int P + \int Q = \int (P \wedge Q) + \int (P \vee Q)$
Dur-Add

$(\int P = x) ; (\int P = y) \implies \int P = x + y$
Dur-Chop

Durations

A Sound Calculus for DC

$$\begin{array}{ll} (1) & \sqcap \implies F \\ (2) & F ; \lceil P \rceil \implies F \\ (3) & F ; \lceil \neg P \rceil \implies F \\ \hline & (4)\ F \end{array}$$
Induction-L

$$\begin{array}{ll} (1) & \sqcap \implies F \\ (2) & \lceil P \rceil ; F \implies F \\ (3) & \lceil \neg P \rceil ; F \implies F \\ \hline & (4)\ F \end{array}$$
Induction-R

Induction

Example

$$\begin{array}{ll} (1) & \sqcap \implies F \\ (2) & F ; \lceil P \rceil \implies F \\ (3) & F ; \lceil \neg P \rceil \implies F \\ \hline & (4)\ F \end{array}$$
Induction-L

$$\begin{array}{ll} (1) & \sqcap \implies F \\ (2) & \lceil P \rceil ; F \implies F \\ (3) & \lceil \neg P \rceil ; F \implies F \\ \hline & (4)\ F \end{array}$$
Induction-R

Let P be a **state assertion** in $E := \sqcap \vee (\mathit{true} ; \lceil P \rceil) \vee (\mathit{true} ; \lceil \neg P \rceil)$

Claim: E is valid.

Proof: Use the **Induction-L** rule.

- (1): obvious
- (2): assume $E ; \lceil P \rceil$.
  - from axiom $E \implies \mathit{true}$, we can derive $(E ; \lceil P \rceil) \implies (\mathit{true} ; \lceil P \rceil)$ by rule **Chop-Mon**

$$\frac{F \implies G}{(F ; H) \implies (G ; H)}$$
Chop-Mon

Example

$$\frac{\begin{array}{ll}(1) & \lceil\,\rceil \implies F\\ (2) & F ; \lceil P \rceil \implies F\\ (3) & F ; \lceil \neg P \rceil \implies F\end{array}}{(4)\ F}\ \textbf{Induction-L} \qquad \frac{\begin{array}{ll}(1) & \lceil\,\rceil \implies F\\ (2) & \lceil P \rceil ; F \implies F\\ (3) & \lceil \neg P \rceil ; F \implies F\end{array}}{(4)\ F}\ \textbf{Induction-R}$$

Let P be a **state assertion** in $E := \lceil\,\rceil \vee (true ; \lceil P \rceil) \vee (true ; \lceil \neg P \rceil)$

Claim: E is valid.

Proof: Use the Induction-L rule.

- (1): obvious
- (2): assume $E ; \lceil P \rceil$.
  - from axiom $E \implies true$, we can derive $(E ; \lceil P \rceil) \implies (true ; \lceil P \rceil)$ by rule Chop-Mon
  - From assumption $(E ; \lceil P \rceil)$, we can derive $(true ; \lceil P \rceil)$ using modus ponens.
  - Thus $E ; \lceil P \rceil \implies E$.
- (3): similar

$$\frac{F \implies G}{(F ; H) \implies (G ; H)}\ \textbf{Chop-Mon} \qquad \frac{F, \quad F \implies G}{G}\ \textbf{modus ponens}$$

Special Cases of Induction

$$\frac{\begin{array}{ll}(1) & \lceil\,\rceil \implies F\\ (2) & F ; \lceil P \rceil \implies F\\ (3) & F ; \lceil \neg P \rceil \implies F\end{array}}{(4)\ F}\ \textbf{Induction-L} \qquad \frac{\begin{array}{ll}(1) & \lceil\,\rceil \implies F\\ (2) & \lceil P \rceil ; F \implies F\\ (3) & \lceil \neg P \rceil ; F \implies F\end{array}}{(4)\ F}\ \textbf{Induction-R}$$

Remark 2.30. For the case $F = (\Box F_1 \implies F_2)$, the premises (2) and (3) of Induction-R can be reduced to

$$(\Box F_1 \wedge F_2 ; \lceil P \rceil) \implies F_2 \qquad (2')$$

$$(\Box F_1 \wedge F_2 ; \lceil \neg P \rceil) \implies F_2 \qquad (3')$$

Special Cases of Induction

$$\frac{\begin{array}{ll}(1) & \lceil\,\rceil \implies F\\ (2) & F ; \lceil P \rceil \implies F\\ (3) & F ; \lceil \neg P \rceil \implies F\end{array}}{(4)\ F}\ \textbf{Induction-L} \qquad \frac{\begin{array}{ll}(1) & \lceil\,\rceil \implies F\\ (2) & \lceil P \rceil ; F \implies F\\ (3) & \lceil \neg P \rceil ; F \implies F\end{array}}{(4)\ F}\ \textbf{Induction-R}$$

Remark 2.31. For the case $F = (\Box F_1 \implies \Box F_2)$, the premises (2) and (3) of Induction-R can be reduced to

$$(\Box F_1 \wedge \Box F_2 ; \lceil P \rceil) \implies F_2 \qquad (2')$$

$$(\Box F_1 \wedge \Box F_2 ; \lceil \neg P \rceil) \implies F_2 \qquad (3')$$

A Complete Calculus for DC?

Theorem 2.23.
A **sound** calculus for DC formulas **cannot** be **complete**.

- Reasons for the necessary incompleteness of sound calculi:
  validity of DC formulae may depend on facts of the real numbers.
  For instance, the fact that every real number is bounded by some natural number (as in the proof of 2.23).
- We only cite: it is impossible to give a complete set of proof rules that characterise all valid facts of the reals.
- What we can have is **relative completeness** in the following sense:

  Given an **"oracle"** for the valid arithmetic formulae over reals, we can always find a proof of F from $\mathcal{H}$.

- The proof system presented earlier is of such a kind.

Content

- **A Calculus for DC**: A brief outlook
  - Recall: **predicate calculus**
  - DC Calculus is **just the same**, just a few more rules
  - → cf. textbook Olderog/Dierks
- **Decidability Results for DC**: Motivation
- **RDC in Discrete Time**
  - Restricted DC **syntax**
  - **Discrete time interpretation** of RDC
  - **Discrete** vs. continuous time
  - The **satisfiability** problem for RDC / discrete time
    - The **language** of a formula

DC Properties

Decidability Results: Motivation

- **Recall**:
  Given **plant assumptions** as a DC formula 'Asm' over the **input observables**, verifying **correctness** of 'Ctrl' wrt. requirements 'Req' amounts to proving

$$\models_0 \text{Ctrl} \wedge \text{Asm} \implies \text{Req} \qquad (1)$$

- If 'Asm' is **not satisfiable** then (1) is trivially valid,
  thus each (!) 'Ctrl' is (trivially) **correct** wrt. 'Req'.
- So: there is a **strong interest** in assessing the **satisfiability** of DC formulae.
- **Question**: is there an automatic procedure to help us out?
  (IOW: is it **decidable** whether a given DC formula is **satisfiable**?)
- Interesting for 'Req': is Req **realisable** (from 0)?
- **Question**: is it **decidable** whether a given DC formula is **realisable**?

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	— " —	— " —

RDC in Discrete Time

Restricted DC (RDC) — Syntax

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

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

First observations (vs. full DC):

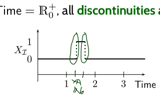
- No global variables (thus don't need $\mathcal{V}$ in semantics).
- Chop operator is there.
- Integral '$\int$' and length 'ℓ'? "Hidden" in $\int P$.
- Predicate and function symbols? No.
- For some subinterval '$\Diamond F$'? In a minute.
- Empty interval '$\lceil\rceil$'? In a minute.

Discrete Time Interpretations of Observables

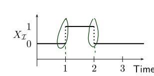
- An interpretation $\mathcal{I}$ is called **discrete time interpretation** if and only if, for each state variable X,

$$X_\mathcal{I} : \text{Time} \to \mathcal{D}(X)$$

with $\text{Time} = \mathbb{R}_0^+$, all **discontinuities are in** $\mathbb{N}_0$.



Not a discrete time interpretation.



A discrete time interpretation.

Discrete Time Interpretation of RDC Formulae

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

- An interval $[b, e] \subset \text{Intv}$ is called **discrete** if and only if $b, e \in \mathbb{N}_0$.
- We say (for a discrete time interpretation $\mathcal{I}$ and a discrete interval $[b, e]$)

$$\mathcal{I}, [b, e] \models F_1 \,;\, F_2$$

if and only if there exists $m \in [b, e] \cap \mathbb{N}_0$ such that

$$\mathcal{I}, [b, m] \models F_1 \quad \text{and} \quad \mathcal{I}, [m, e] \models F_2$$

- The interpretations of '$\vee$' and '$\neg$' remain unchanged.
- $\mathcal{I}, [b, e] \models \lceil P \rceil$ if and only if $\int_b^e P_\mathcal{I}(t)\, dt = (e - b)$ and $e - b > 0$.

Differences between Continuous and Discrete Time

- Let P be a state assertion.

smallest $e-b$: 2

	Continuous Time	Discrete Time
$\models^? (\lceil P \rceil ; \lceil P \rceil) \implies \lceil P \rceil$	yes ✓ //// NO ✗ -	✓ yes ✗ NO ////
$\models^? \lceil P \rceil \implies (\lceil P \rceil ; \lceil P \rceil)$	yes ✓ 𝍸/ NO ✗ -	✓ YES - ✗ NO //

b $\lceil P \rceil$ $e \geq b$ $e = b+1$

smallest: $e-b = 1$

Differences between Continuous and Discrete Time

- Let P be a state assertion.

	Continuous Time	Discrete Time
$\models^? (\lceil P \rceil ; \lceil P \rceil) \implies \lceil P \rceil$	✔	✔
$\models^? \lceil P \rceil \implies (\lceil P \rceil ; \lceil P \rceil)$	✔	✘

- In particular: $\ell = 1 \iff (\lceil 1 \rceil \wedge \neg(\lceil 1 \rceil ; \lceil 1 \rceil))$ (in discrete time).

Expressiveness of RDC

- $\ell = 1 \iff \lceil 1 \rceil \wedge \neg(\lceil 1 \rceil ; \lceil 1 \rceil)$
- $\ell = 0 \iff \neg \lceil 1 \rceil$
- $true \iff \ell = 0 \vee \neg(\ell = 0)$
- $\int P = 0 \iff \lceil \neg P \rceil \vee \ell = 0$
- $\int P = 1 \iff (\int P = 0) ; (\lceil P \rceil \wedge \ell = 1) ; (\int P = 0)$
- $\int P = k+1 \iff (\int P = k) ; (\int P = 1)$
- $\int P \geq k \iff (\int P = k) ; true$
- $\int P > k \iff \int P \geq k+1$
- $\int P \leq k \iff \neg(\int P > k)$
- $\int P < k \iff \int P \leq k-1$

where $k \in \mathbb{N}$.

$\Diamond F :=$ true ; F ; true

Decidability Results for RDC in Discrete Time

Decidability of Satisfiability/Realisability from 0

Theorem 3.6.
The satisfiability problem for RDC with discrete time is decidable.

Theorem 3.9.
The realisability problem for RDC with discrete time is decidable.

Sketch: Proof of Theorem 3.6

- Give a procedure to construct, given a formula F, a **regular** language $\mathcal{L}(F)$ such that

$$\mathcal{I}, [0, n] \models F \text{ if and only if } w \in \mathcal{L}(F)$$

where word w describes $\mathcal{I}$ on $[0, n]$
(suitability of the procedure: Lemma 3.4).

- Then F is satisfiable in discrete time if and only if $\mathcal{L}(F)$ is not empty (Lemma 3.5).

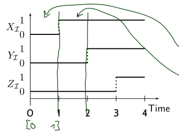
- Theorem 3.6 follows because
  - $\mathcal{L}(F)$ can **effectively** be constructed,
  - the emptyness problem is **decidable** for regular languages.

Alphabet of a Formula

- **Idea:**
  - **alphabet** $\Sigma(F)$ consists of **basic conjuncts** of the state variables in F.
  - a **letter** corresponds to an **interpretation on an interval of length 1**.
  - a **word** of length n describes an interpretation on interval $[0, n]$.

- **Example:** Assume F contains exactly state variables X, Y, Z, then

$$\Sigma(F) = \{X \wedge Y \wedge Z, \quad X \wedge Y \wedge \neg Z, \quad X \wedge \neg Y \wedge Z, \quad X \wedge \neg Y \wedge \neg Z,$$
$$\neg X \wedge Y \wedge Z, \quad \neg X \wedge Y \wedge \neg Z, \quad \neg X \wedge \neg Y \wedge Z, \quad \neg X \wedge \neg Y \wedge \neg Z\}.$$



$$w = (\neg X \wedge \neg Y \wedge \neg Z) \cdot (X \wedge \neg Y \wedge \neg Z) \cdot (X \wedge Y \wedge \neg Z) \cdot (X \wedge Y \wedge Z) \in \Sigma(F)^*$$

$\Sigma(F)$

Words vs. Interpretations

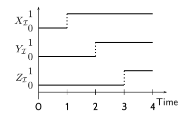
Definition 3.2. A word $w = a_1 \dots a_n \in \Sigma(F)^*$ with $n \geq 0$ **describes** a **discrete** interpretation $\mathcal{I}$ on $[0, n]$ if and only if

$$\forall j \in \{1, \dots, n\} \, \forall t \in]j-1, j[: \mathcal{I}[\![a_j]\!](t) = 1.$$

For $n = 0$ we set $w = \varepsilon$.

- **Example:** word w **describes** $\mathcal{I}$ on $[0, 4]$.

$$\Sigma(F) = \{X \wedge Y \wedge Z, \quad X \wedge Y \wedge \neg Z, \quad X \wedge \neg Y \wedge Z, \quad X \wedge \neg Y \wedge \neg Z,$$
$$\neg X \wedge Y \wedge Z, \quad \neg X \wedge Y \wedge \neg Z, \quad \neg X \wedge \neg Y \wedge Z, \quad \neg X \wedge \neg Y \wedge \neg Z\}.$$



$$w = (\neg X \wedge \neg Y \wedge \neg Z) \cdot (X \wedge \neg Y \wedge \neg Z) \cdot (X \wedge Y \wedge \neg Z) \cdot (X \wedge Y \wedge Z) \in \Sigma(F)^*$$

Construction of the Language $\mathcal{L}(F)$ of Formula F

- Note: Each state assertion P can be transformed into an equivalent **disjunctive normal form** $\bigvee_{i=1}^m a_i$ with $a_i \in \Sigma(F)$.

$P = X \wedge \neg Y \iff X \wedge \neg Y \wedge Z \vee X \wedge \neg Y \wedge \neg Z$ } $DNF(P)$

- Set $DNF(P) := \{a_1, \dots, a_m\}$ $(\subseteq \Sigma(F))$.

- Define $\mathcal{L}(F)$ inductively:

finitely many, at least one

$$\mathcal{L}(\lceil P \rceil) = DNF(P)^+$$
$$\mathcal{L}(\neg F_1) = \Sigma(F)^* \setminus \mathcal{L}(F_1)$$
$$\mathcal{L}(F_1 \vee F_2) = \mathcal{L}(F_1) \cup \mathcal{L}(F_2)$$
$$\mathcal{L}(F_1 \,;\, F_2) = \mathcal{L}(F_1) \cdot \mathcal{L}(F_2)$$

Construction of the Language $\mathcal{L}(F)$ of Formula F

- Note: Each state assertion P can be transformed into an equivalent **disjunctive normal form** $\bigvee_{i=1}^m a_i$ with $a_i \in \Sigma(F)$.

- Set $DNF(P) := \{a_1, \dots, a_m\}$ $(\subseteq \Sigma(F))$.

- Define $\mathcal{L}(F)$ inductively:

$$\mathcal{L}(\lceil P \rceil) = DNF(P)^+,$$
$$\mathcal{L}(\neg F_1) = \Sigma(F)^* \setminus \mathcal{L}(F_1),$$
$$\mathcal{L}(F_1 \vee F_2) = \mathcal{L}(F_1) \cup \mathcal{L}(F_2),$$
$$\mathcal{L}(F_1 \,;\, F_2) = \mathcal{L}(F_1) \cdot \mathcal{L}(F_2).$$

Lemma 3.4

Lemma 3.4. For all RDC formulae F, discrete interpretations $\mathcal{I}$, $n \geq 0$, and all words $w \in \Sigma(F)^*$ which **describe** $\mathcal{I}$ on $[0, n]$,

$$\mathcal{I}, [0, n] \models F \text{ if and only if } w \in \mathcal{L}(F).$$

Proof: By structural induction.

- **Base case:** $F = \lceil P \rceil$:
  - Let $w = a_1, \dots, a_n$, $n \geq 0$, **describe** $\mathcal{I}$ on $[0, n]$.
  - $\mathcal{I}, [0, n] \models \lceil P \rceil$

$\iff \mathcal{I}, [0, n] \models \lceil P \rceil$ and $n \geq 1$

$\iff n \geq 1$ and $\forall 1 \leq j \leq n \bullet \mathcal{I}, [j-1, j] \models \lceil P \rceil$

$\iff n \geq 1$ and $\forall 1 \leq j \leq n \bullet \mathcal{I}, [j-1, j] \models (\lceil P \rceil \wedge \lceil a_j \rceil)$ and $a_j \in DNF(P)$

$\iff n \geq 1$ and $\forall 1 \leq j \leq n \bullet a_j \in DNF(P)$

$\iff w \in DNF(P)^+ \iff w \in \mathcal{L}(F)$

Lemma 3.4 Cont'd

Lemma 3.4. For all RDC formulae F, discrete interpretations $\mathcal{I}$, $n \geq 0$, and all words $w \in \Sigma(F)^*$ which **describe** $\mathcal{I}$ on $[0, n]$,

$$\mathcal{I}, [0, n] \models F \text{ if and only if } w \in \mathcal{L}(F).$$

Proof: By structural induction.

- **Induction steps:** $F = \neg F_1$:
  - Let $w = a_1, \dots, a_n$, $n \geq 0$, **describe** $\mathcal{I}$ on $[0, n]$.
  - $\mathcal{I}, [0, n] \models \neg F_1$

$\iff$ not $\mathcal{I}, [0, n] \models F_1$

$\iff w \notin \mathcal{L}(F_1)$

$\iff w \in \overline{\mathcal{L}(F_1)}$

$\iff w \in \mathcal{L}(\neg F_1)$ by def.

- $F_1 \vee F_2$, $F_1 \,;\, F_2$: similar

Sketch: Proof of Theorem 3.9

Theorem 3.9.
The realisability problem for RDC with discrete time is decidable.

- $kern(L)$ contains all words of L whose prefixes are again in L.
- If L is regular, then $kern(L)$ is also regular.
- $kern(\mathcal{L}(F))$ can effectively be constructed.

- We have

Lemma 3.8. For all RDC formulae F, F is realisable from 0 in discrete time if and only if $kern(\mathcal{L}(F))$ is infinite.

- Infinity of regular languages is decidable.

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	— " —	— " —

Content

Tell Them What You've Told Them...

- A **sound calculus** for DC exists, a **complete** calculus does not exist.

  Knowing the (sound) proof rules may also be useful when conducting **correctness proofs** manually.

  → see the textbook for the details

- **Decidability** of, e.g., satisfiability of DC formulae is **interesting**.

  A decision procedure could analyse, e.g., whether plant assumptions Asm are (at least) satisfiable.

- For **Restricted DC** in **discrete time**,
  - **satisfiability** is **decidable**.
  - **Proof idea**: reduce to regular languages.

References

References

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