Real-Time Systems

Lecture 05: Duration Calculus III

2014-05-20

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
Contents & Goals

Last Lecture:

- DC Syntax and Semantics: Formulae

This Lecture:

- Educational Objectives: Capabilities for following tasks/questions.
  - Read (and at best also write) Duration Calculus formulae – including abbreviations.
  - What is Validity/Satisfiability/Realisability for DC formulae?
  - How can we prove a design correct?

- Content:
  - Duration Calculus Abbreviations
  - Basic Properties
  - Validity, Satisfiability, Realisability
  - Correctness Proofs: Gas Burner
Duration Calculus Abbreviations
Abbreviations

- $[] := \ell = 0$ (point interval)
- $[P] := \int P = \ell \land \ell > 0$ (almost everywhere)
- $[P]^t := [P] \land \ell = t$ (for time $t$)
- $[P]^{\leq t} := [P] \land \ell \leq t$ (up to time $t$)
- $\Diamond F := true ; F ; true$ (for some subinterval)
- $\Box F := \neg \Diamond \neg F$ (for all subintervals)
Abbreviations: Examples

\[
\begin{align*}
\mathcal{I}[\int L = 0] &= (\mathcal{V}, [0, 2]) = \\
\mathcal{I}[\int L = 1] &= (\mathcal{V}, [2, 6]) = \\
\mathcal{I}[\int L = 0; \int L = 1] &= (\mathcal{V}, [0, 6]) = \\
\mathcal{I}[\neg L] &= (\mathcal{V}, [0, 2]) = \\
\mathcal{I}[L] &= (\mathcal{V}, [2, 3]) = \\
\mathcal{I}[\neg L ; [L]] &= (\mathcal{V}, [0, 3]) = \\
\mathcal{I}[\neg L ; [L]; [\neg L]] &= (\mathcal{V}, [0, 6]) = \\
\mathcal{I}[\lozenge [L]] &= (\mathcal{V}, [0, 6]) = \\
\mathcal{I}[\lozenge [\neg L]] &= (\mathcal{V}, [0, 6]) = \\
\mathcal{I}[\lozenge [\neg L]^2] &= (\mathcal{V}, [0, 6]) = \\
\mathcal{I}[\lozenge [\neg L]^2; [\neg L]^1; [\neg L]^3] &= (\mathcal{V}, [0, 6]) =
\end{align*}
\]
Duration Calculus: Looking Back

- Duration Calculus is an interval logic.
- Formulae are evaluated in an (implicitly given) interval.

Strangest operators:

- **almost everywhere** — Example: \([G]\)
  (Holds in a given interval \([b, e]\) iff the gas valve is open almost everywhere.)

- **chop** — Example: \((\lceil \neg I \rceil ; \lceil I \rceil ; \lceil \neg I \rceil) \implies \ell \geq 1\)
  (Ignition phases last at least one time unit.)

- **integral** — Example: \(\ell \geq 60 \implies \int L \leq \frac{\ell}{20}\)
  (At most 5% leakage time within intervals of at least 60 time units.)

- \(G, F, I, H : \{0, 1\}\)
- Define \(L : \{0, 1\}\) as \(G \land \neg F\).
DC Validity, Satisfiability, Realisability
Let $\mathcal{I}$ be an interpretation, $\mathcal{V}$ a valuation, $[b, e]$ an interval, and $F$ a DC formula.

- $\mathcal{I}, \mathcal{V}, [b, e] \models F$ ("$F$ holds in $\mathcal{I}, \mathcal{V}, [b, e]$") iff $\mathcal{I}[F](\mathcal{V}, [b, e]) = \text{tt}$. 
Validity, Satisfiability, Realisability

Let $\mathcal{I}$ be an interpretation, $\mathcal{V}$ a valuation, $[b, e]$ an interval, and $F$ a DC formula.

- $\mathcal{I}, \mathcal{V}, [b, e] \models F$ ("$F$ holds in $\mathcal{I}, \mathcal{V}, [b, e]$") iff $\mathcal{I}[F](\mathcal{V}, [b, e]) = \text{tt}$.

- $F$ is called **satisfiable** iff it holds in some $\mathcal{I}, \mathcal{V}, [b, e]$. 
Validity, Satisfiability, Realisability

Let $\mathcal{I}$ be an interpretation, $\mathcal{V}$ a valuation, $[b, e]$ an interval, and $F$ a DC formula.

- $\mathcal{I}, \mathcal{V}, [b, e] \models F$ ("$F$ holds in $\mathcal{I}, \mathcal{V}, [b, e]$") iff $\mathcal{I}[F](\mathcal{V}, [b, e]) = \text{tt}.$

- $F$ is called **satisfiable** iff it holds in some $\mathcal{I}, \mathcal{V}, [b, e].$

- $\mathcal{I}, \mathcal{V} \models F$ ("$\mathcal{I}$ and $\mathcal{V}$ realise $F$") iff $\forall [b, e] \in \text{Intv} : \mathcal{I}, \mathcal{V}, [b, e] \models F.$
Validity, Satisfiability, Realisability

Let $\mathcal{I}$ be an interpretation, $\mathcal{V}$ a valuation, $[b, e]$ an interval, and $F$ a DC formula.

- $\mathcal{I}, \mathcal{V}, [b, e] \models F$ ("$F$ holds in $\mathcal{I}, \mathcal{V}, [b, e]$") iff $\mathcal{I}[F](\mathcal{V}, [b, e]) = \text{tt}$.

- $F$ is called **satisfiable** iff it holds in some $\mathcal{I}, \mathcal{V}, [b, e]$.

- $\mathcal{I}, \mathcal{V} \models F$ ("$\mathcal{I}$ and $\mathcal{V}$ realise $F$"") iff $\forall [b, e] \in \text{Intv} : \mathcal{I}, \mathcal{V}, [b, e] \models F$.

- $F$ is called **realisable** iff some $\mathcal{I}$ and $\mathcal{V}$ realise $F$.
### Validity, Satisfiability, Realisability

Let $\mathcal{I}$ be an interpretation, $\mathcal{V}$ a valuation, $[b, e]$ an interval, and $F$ a DC formula.

- $\mathcal{I}, \mathcal{V}, [b, e] \models F$ ("$F$ holds in $\mathcal{I}, \mathcal{V}, [b, e]$") iff $\mathcal{I}[F](\mathcal{V}, [b, e]) = \text{tt}$.

- $F$ is called **satisfiable** iff it holds in some $\mathcal{I}, \mathcal{V}, [b, e]$.

- $\mathcal{I}, \mathcal{V} \models F$ ("$\mathcal{I}$ and $\mathcal{V}$ realise $F$") iff $\forall [b, e] \in \text{Intv} : \mathcal{I}, \mathcal{V}, [b, e] \models F$.

- $F$ is called **realisable** iff some $\mathcal{I}$ and $\mathcal{V}$ realise $F$.

- $\mathcal{I} \models F$ ("$\mathcal{I}$ realises $F$") iff $\forall \mathcal{V} \in \text{Val} : \mathcal{I}, \mathcal{V} \models F$.
Let \( \mathcal{I} \) be an interpretation, \( \mathcal{V} \) a valuation, \([b, e]\) an interval, and \( F \) a DC formula.

- \( \mathcal{I}, \mathcal{V}, [b, e] \models F \) ("\( F \) holds in \( \mathcal{I}, \mathcal{V}, [b, e] \)") iff \( \mathcal{I}[F](\mathcal{V}, [b, e]) = \text{tt} \).

- \( F \) is called **satisfiable** iff it holds in some \( \mathcal{I}, \mathcal{V}, [b, e] \).

- \( \mathcal{I}, \mathcal{V} \models F \) ("\( \mathcal{I} \) and \( \mathcal{V} \) realise \( F \)") iff \( \forall [b, e] \in \text{Intv} : \mathcal{I}, \mathcal{V}, [b, e] \models F \).

- \( F \) is called **realisable** iff some \( \mathcal{I} \) and \( \mathcal{V} \) realise \( F \).

- \( \mathcal{I} \models F \) ("\( \mathcal{I} \) realises \( F \)") iff \( \forall \mathcal{V} \in \text{Val} : \mathcal{I}, \mathcal{V} \models F \).

- \( \models F \) ("\( F \) is valid") iff \( \forall \) interpretation \( \mathcal{I} : \mathcal{I} \models F \).
Remark 2.13. For all DC formulae $F$,

- $F$ is satisfiable iff $\neg F$ is not valid,
  $F$ is valid iff $\neg F$ is not satisfiable.
- If $F$ is valid then $F$ is realisable, but not vice versa.
- If $F$ is realisable then $F$ is satisfiable, but not vice versa.
Examples: Valid? Realisable? Satisfiable?

- \( \ell \geq 0 \)
- \( \ell = \int 1 \)
- \( \ell = 30 \iff \ell = 10 ; \ell = 20 \)
- \( ((F ; G) ; H) \iff (F ; (G ; H)) \)
- \( \int L \leq x \)
- \( \ell = 2 \)
Initial Values

- $\mathcal{I}, \mathcal{V} \models_0 F$ ("$\mathcal{I}$ and $\mathcal{V}$ realise $F$ from 0") iff
  \[ \forall t \in \text{Time} : \mathcal{I}, \mathcal{V}, [0, t] \models F. \]

- $F$ is called realisable from 0 iff some $\mathcal{I}$ and $\mathcal{V}$ realise $F$ from 0.

- Intervals of the form $[0, t]$ are called initial intervals.

- $\mathcal{I} \models_0 F$ ("$\mathcal{I}$ realises $F$ from 0") iff
  \[ \forall \mathcal{V} \in \text{Val} : \mathcal{I}, \mathcal{V} \models_0 F. \]

- $\models_0 F$ ("$F$ is valid from 0") iff
  \[ \forall \text{interpretation } \mathcal{I} : \mathcal{I} \models_0 F. \]
Initial or not Initial...

For all interpretations $\mathcal{I}$, valuations $\mathcal{V}$, and DC formulae $F$,

(i) $\mathcal{I}, \mathcal{V} \models F$ implies $\mathcal{I}, \mathcal{V} \models_0 F$,

(ii) if $F$ is realisable then $F$ is realisable from 0, but not vice versa,

(iii) $F$ is valid iff $F$ is valid from 0.
Specification and Semantics-based Correctness Proofs of Real-Time Systems with DC
(i) Choose a collection of observables ‘Obs’.

(ii) Provide the requirement/specification ‘Spec’ as a conjunction of DC formulae (over ‘Obs’).

(iii) Provide a description ‘Ctrl’ of the controller in form of a DC formula (over ‘Obs’).

(iv) We say ‘Ctrl’ is correct (wrt. ‘Spec’) iff

\[ \models_0 \text{Ctrl} \implies \text{Spec}. \]
(i) Choose observables:

- two boolean observables $G$ and $F$
  (i.e. $\text{Obs} = \{G, F\}$, $\mathcal{D}(G) = \mathcal{D}(F) = \{0, 1\}$)
- $G = 1$: gas valve open
- $F = 1$: have flame
- define $L := G \land \neg F$ (leakage)

(ii) Provide the requirement:

$$\text{Req} : \iff \Box(\ell \geq 60 \implies 20 \cdot \int L \leq \ell)$$
(iii) Provide a description ‘Ctrl’ of the controller in form of a DC formula (over ‘Obs’). Here, firstly consider a design:

- Des-1: $\iff \Box([L] \implies \ell \leq 1)$
- Des-2: $\iff \Box([L] ; [\neg L] ; [L] \implies \ell > 30)$

(iv) Prove correctness:

- We want (or do we want $|=_{0...}$):

  $|= (\text{Des-1} \land \text{Des-2} \implies \text{Req})$  \hspace{1cm} (Thm. 2.16)
(iii) Provide a description ‘Ctrl’ of the controller in form of a DC formula (over ‘Obs’). Here, firstly consider a design:

- Des-1 : ⇐⇒ □([L] ⇒ ℓ ≤ 1)


(iv) Prove correctness:

- We want (or do we want |=₀…?):

  |= (Des-1 ∧ Des-2 ⇒ Req)  \hspace{1cm} (Thm. 2.16)

- We do show

  |= Req-1 ⇒ Req  \hspace{1cm} (Lem. 2.17)

  with the simplified requirement

  Req-1 := □(ℓ ≤ 30 ⇒ ∫L ≤ 1),
Claim:

\[ \models \square(\ell \leq 30 \implies \int L \leq 1) \implies \square(\ell \geq 60 \implies 20 \cdot \int L \leq \ell) \]

Proof:
Claim:

\[ \vdash \Box (\ell \leq 30 \implies \int L \leq 1) \implies \Box (\ell \geq 60 \implies 20 \cdot \int L \leq \ell) \]

Proof:

- Assume ‘Req-1’.
Claim:

\[ \models \square(\ell \leq 30 \implies \int L \leq 1) \implies \square(\ell \geq 60 \implies 20 \cdot \int L \leq \ell) \]

Proof:

- Assume ‘Req-1’.
- Let \( L_\mathcal{I} \) be any interpretation of \( L \), and \([b, e]\) an interval with \( e - b \geq 60 \).
Claim:

\[ \models \Box (\ell \leq 30 \implies \int L \leq 1) \implies \Box (\ell \geq 60 \implies 20 \cdot \int L \leq \ell) \]

Proof:

- Assume ‘Req-1’.
- Let \( L_I \) be any interpretation of \( L \), and \([b, e]\) an interval with \( e - b \geq 60 \).
- Show “\( 20 \cdot \int L \leq \ell \)”, i.e.

\[ \mathcal{I}[20 \cdot \int L \leq \ell]((\forall, [b, e])) = \text{tt} \]

i.e.

\[ 20 \cdot \int_b^e L_I(t) \, dt \leq (e - b) \]
Set \( n := \lceil \frac{e-b}{30} \rceil \), i.e. \( n \in \mathbb{N} \) with \( n-1 < \frac{e-b}{30} \leq n \), and split the interval

\[
b + 30 \quad b + 60 \quad b + 30(n-2)b + 30(n-1) \quad b + 30n
\]

\[
b \quad \text{.........} \quad e
\]
Some Laws of the DC Integral Operator

**Theorem 2.18.**
For all state assertions $P$ and all real numbers $r_1, r_2 \in \mathbb{R}$,

(i) $\models \int P \leq \ell$,
(ii) $\models (\int P = r_1) \land (\int P = r_2) \implies \int P = r_1 + r_2$,
(iii) $\models \lceil \neg P \rceil \implies \int P = 0$,
(iv) $\models \lceil \neg \rceil \implies \int P = 0$. 
Claim:

\[ \models (\Box ([L] \implies \ell \leq 1) \land \Box ([L] ; [\neg L] ; [L] \implies \ell > 30)) \implies \Box (\ell \leq 30 \implies \int L \leq 1) \]

Proof:
Claim:

\[ \vdash (\square([L] \implies \ell \leq 1) \land \square([L] ; [\neg L] ; [L] \implies \ell > 30)) \implies \square(\ell \leq 30 \implies \ell L \leq 1) \]

Proof:

\( (i) \models \int P \leq \ell, \quad (iv) \models \square \implies \int P = 0 \)

\( (ii) \models (\int P = r_1) ; (\int P = r_2) \implies \int P = r_1 + r_2 \)

\( (iii) \models [\neg P] \implies \int P = 0 \)
Obstacles in Non-Ideal World
Methodology: The World is Not Ideal...

(i) Choose a collection of **observables** ‘Obs’.

(ii) Provide **specification** ‘Spec’ (conjunction of DC formulae (over ‘Obs’)).

(iii) Provide a description ‘Ctrl’ of the **controller** (DC formula (over ‘Obs’)).

(iv) Prove ‘Ctrl’ is **correct** (wrt. ‘Spec’).

That looks **too simple to be practical**. Typical **obstacles**:

(i) It may be impossible to realise ‘Spec’ if it doesn’t consider properties of the **plant**.

(ii) There are typically intermediate **design levels** between ‘Spec’ and ‘Ctrl’.

(iii) ‘Spec’ and ‘Ctrl’ may use **different observables**.

(iv) **Proving** validity of the implication is not trivial.
(i) Assumptions As A Form of Plant Model

- Often the controller will (or can) operate correctly only under some assumptions.
- For instance, with a level crossing
  - we may assume an upper bound on the speed of approaching trains, (otherwise we’d need to close the gates arbitrarily fast)
  - we may assume that trains are not arbitrarily slow in the crossing, (otherwise we can’t make promises to the road traffic)
Often the controller will (or can) operate correctly only under some assumptions.

For instance, with a level crossing

- we may assume an upper bound on the speed of approaching trains, (otherwise we’d need to close the gates arbitrarily fast)
- we may assume that trains are not arbitrarily slow in the crossing, (otherwise we can’t make promises to the road traffic)

We shall specify such assumptions as a DC formula ‘Asm’ on the input observables and verify correctness of ‘Ctrl’ wrt. ‘Spec’ by proving validity (from 0) of

\[ \text{Ctrl} \land \text{Asm} \implies \text{Spec} \]
(i) Assumptions As A Form of Plant Model

- Often the controller will (or can) operate correctly only under some assumptions.

- For instance, with a level crossing
  - we may assume an upper bound on the speed of approaching trains, (otherwise we’d need to close the gates arbitrarily fast)
  - we may assume that trains are not arbitrarily slow in the crossing, (otherwise we can’t make promises to the road traffic)

- We shall specify such assumptions as a DC formula ‘Asm’ on the input observables and verify correctness of ‘Ctrl’ wrt. ‘Spec’ by proving validity (from 0) of

  \[ \text{Ctrl} \wedge \text{Asm} \implies \text{Spec} \]

- Shall we care whether ‘Asm’ is satisfiable?
(ii) Intermediate Design Levels

- A top-down development approach may involve
  - Spec — specification/requirements
  - Des — design
  - Ctrl — implementation

- Then correctness is established by proving validity of

  \[ \text{Ctrl} \implies \text{Des} \quad (1) \]

  and

  \[ \text{Des} \implies \text{Spec} \quad (2) \]

  (then concluding \( \text{Ctrl} \implies \text{Spec} \) by transitivity)

- Any preference on the order?
(iii): Different Observables

- Assume, ‘Spec’ uses more abstract observables $\text{Obs}_A$ and ‘Ctrl’ more concrete ones $\text{Obs}_C$.

- For instance:
  - in $\text{Obs}_A$: only consider gas valve open or closed ($\mathcal{D}(G) = \{0, 1\}$)
  - in $\text{Obs}_C$: may control two valves and care for intermediate positions, for instance, to react to different heating requests ($\mathcal{D}(G_1) = \{0, 1, 2, 3\}, \mathcal{D}(G_2) = \{0, 1, 2, 3\}$)
(iii): Different Observables

- Assume, ‘Spec’ uses more abstract observables $\text{Obs}_A$ and ‘Ctrl’ more concrete ones $\text{Obs}_C$.

- For instance:
  - in $\text{Obs}_A$: only consider gas valve open or closed ($\mathcal{D}(G) = \{0, 1\}$)
  - in $\text{Obs}_C$: may control two valves and care for intermediate positions, for instance, to react to different heating requests ($\mathcal{D}(G_1) = \{0, 1, 2, 3\}, \mathcal{D}(G_2) = \{0, 1, 2, 3\}$)

- To prove correctness, we need information how the observables are related — an invariant which links the data values of $\text{Obs}_A$ and $\text{Obs}_C$.

- If we’re given the linking invariant as a DC formula, say ‘$\text{Link}_{C,A}$’, then proving correctness of ‘Ctrl’ wrt. ‘Spec’ amounts to proving validity (from 0) of

$$\text{Ctrl} \land \text{Link}_{C,A} \implies \text{Spec}. $$

- For instance,

$$\text{Link}_{C,A} = \left[ G \iff (G_1 + G_2 > 0) \right]$$
Obstacle (iv): How to Prove Correctness?

- by hand on the basis of DC semantics,
- maybe supported by proof rules,
- sometimes a general theorem may fit (e.g. cycle times of PLC automata),
- algorithms as in Uppaal.
Recall: Tying It All Together

**abstraction level**

**formal description language I**

- **Duration Calculus**
- **Constraint Diagrams**

**semantic integration**

- **DC**

**automatic verification**

- **timed automata**

**formal descr. language II**

- **Live Seq. Charts**

**Requirements**

- **satisfied by**

**Designs**

- **PLC-Automata**

**Programs**

- **C code**
  - PLC code

**compiler**

- **equiv.**

- **equiv.**

- **equiv.**

- **equiv.**
References