Duration Calculus: Preview

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

Strangest operators:
• almost everywhere — Example: \( \left( [x_1, x_2]: [y_1, y_2] \right) \) holds in a given interval \( [x_1, x_2] \) if the gas valve is open almost everywhere.
• chop — Example:
  \[
  \left( [x_1, x_2]; [y_1, y_2]; [z_1, z_2] \right) = [z_1, z_2] \text{ (ignition phases last at least one time unit.)}
  \]

Integral — Example: at most 5% leakage time within intervals of at least 60 time units.

\[
\int_{x_1}^{x_2} \leq \frac{5}{100} \Rightarrow \int_{y_1}^{y_2} \leq 60 \quad \text{ iff } \quad \int_{y_1}^{y_2} \leq 60
\]

Define \( \Delta T \):
\[
\Delta T = L = \left\{ 0, 1 \right\}
\]

Integral operation:

\[
\int_{x_1}^{x_2} 60 = \int_{y_1}^{y_2} L
\]

(At most 5% leakage time within intervals of at least 60 time units.)
• 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

Introduction

• Requirements
  • 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 . . .

• Semantics-based Correctness Proofs
• Example: Gas Burner Controller
• Theorem 2.16:
  Des-1 and Des-2 is a correct design wrt. Req
• Lemma 2.19:
  Des-1 and Des-2 imply a simplified requirement Req-1

• Obstacles (in a non-ideal world)
  • requirements may be unrealisable
  • intermediate design levels
  • different observables
  • proving correctness may be difficult

• Example Gas Burner Control
In order to prove a controller design correct wrt a specification:

\[ \text{Methodology (in an ideal world)} \]

\[ \text{Step }, 1 \text{: Choose observables, } \mathcal{O}_\text{obs} \]

\[ \text{Step }, 2 \text{: Formalise the requirements, } \mathcal{R}_\text{req} \]

\[ \text{Step }, 3 \text{: Formalise a controller design, } \mathcal{C}_\text{ctrl} \]

\[ \text{Step }, 4 \text{: As a conjunction of DC formulae (over } \mathcal{O}_\text{obs}) \]

\[ \text{Step }, 5 \text{: We say } \mathcal{C}_\text{ctrl} \text{ is correct wrt } \mathcal{R}_\text{req} \text{ iff } \]

\[ \vdash \mathcal{R}_\text{req} \Rightarrow \mathcal{C}_\text{ctrl} \]

⇒ \[ \text{So just prove } \vdash \mathcal{C}_\text{ctrl} \text{ to } \mathcal{R}_\text{req} \text{.} \]
Gas Burner Revisited

(i) Choose observables:
• $F$: \{0, 1\}: value 1 models "flame sensed now" (input)
• $G$: \{0, 1\}: value 1 models "gas valve is open now" (output)

(ii) Formalise the requirement:
$\text{Req} := \Box (\ell \geq 60 = \Rightarrow 20 \cdot \int L \leq \ell)$
"in each interval of length at least 60 time units, at most 5% of the time"

(iii) Formalise controller design ideas:
• $\text{Des}_1 := \Box (\lceil L \rceil = \Rightarrow \ell \leq 1)$ "make leakage phases last for at most one time unit"
• $\text{Des}_2 := \Box (\lceil L \rceil; \lceil \neg L \rceil; \lceil L \rceil = \Rightarrow \ell > 30)$ "ensure non-leakage phases between two leakage phases last at least 30 time units"

(iv) Prove correctness, i.e., prove $|\text{Des}_1 \land \text{Des}_2 = \Rightarrow \text{Req})$.
(Or do we want "$|\text{Des}_1 = 0$ ...?)
A Correct Gas Burner Controller Design

\[(q - e) \leq t \Rightarrow (\ell \leq 60 = \Rightarrow 20 \cdot \int L \leq \ell)\]

\[\text{Des-1} := \Box (\lceil L \rceil = \Rightarrow \ell \leq 1), \quad \text{Des-2} := \Box (\lceil L \rceil; \lceil \neg L \rceil; \lceil L \rceil = \Rightarrow \Box)\]

A controller for the gas burner which guarantees Des-1 and Des-2 is correct.

\[\text{Des-1} \land \text{Des-2} \Rightarrow \Box (\ell \geq 60 = \Rightarrow 20 \cdot \int L \leq \ell)\]

We have \((\text{n= lemma 2.17})\) completes the overall proof.

Claim: \(\Box (\ell \leq 30 = \Rightarrow \int L \leq 1)\)

\[\text{Proof:}\]

\[\text{Assume that '}\Box (\ell \leq 30 = \Rightarrow \int L \leq 1)' holds.}\]

\[\text{Let } L \text{ be any interpretation of } L \text{ on interval } [q, e] \text{ and any interval } [b, e] \text{ with } e - b \geq 20.\]

\[\text{We need to show that } 20 \cdot \int L \leq \ell \text{ evaluates to 'tt' on interval } [b, e] \text{ under interpretation } I \text{ (and any valuation V).}\]

\[\text{We have } I[20 \cdot \int L] = \text{tt} \iff (\text{by DC semantics})\]

\[\hat{\hat{20}} \cdot \int_{b}^{e} L I(t) dt \leq (e - b).\]

\[\text{Lemma 2.17}\]

Showing \(\text{for the simplified requirement}\) \(\Box (\ell \geq 60 = \Rightarrow 20 \cdot \int L \leq \ell)\)

\[\text{We do prove (in Lemma 2.19) (\text{shown in theorem 2.16})}\]

\[\text{A Correct Gas Burner Controller Design}\]
Lemma 2.17 Cont'd

\[ 0 + \frac{(q - a)}{\varepsilon} = \left(1 + \frac{20}{q - a}\right) \cdot 0 \geq \{a\} \]

\[ u \cdot 20 \leq 20 + \frac{(q - a)}{\varepsilon} \quad \{90 \leq q - a\} \]

\[ \int_{0 \leq \varepsilon} (1 - u) \cdot 90 + q + q = q \]

\[ \int_{0 \leq \varepsilon} (1 - u) \cdot 90 + q + q = q \]

Set \( n \leq \frac{q - a}{\varepsilon} \geq 1 - I \) with \( n \in \mathbb{N} \) that \( n = \varepsilon \) \(*\)

\[ \{90 \leq q - a\} = \{a\} \]

Lemma 2.17 Cont'd
Theorem 2.18.

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

(i) $|P| = \int P \leq \ell$,

(ii) $|P| = (\int P = r_1)$ \quad \implies \quad \int P = r_1 + r_2$,

(iii) $|\neg P| = \lceil \neg P \rceil \implies \int P = 0$,

(iv) $|\rceil| = \lceil \rceil = \lceil \rceil \implies \int P = 0$.

Lemma 2.19

(i) $|P| = \int P \leq \ell$,

(ii) $|P| = (\int P = r_1)$ \quad \implies \quad \int P = r_1 + r_2$,

(iii) $|\neg P| = \lceil \neg P \rceil \implies \int P = 0$.

Claim:

$\{\text{Des-1} \land \text{Des-2}\} \land (\lceil L \rceil \implies \ell \leq 3 \land \neg \text{Des-2}) \implies \neg \text{Des-2}$

Proof:

$\ell \leq 30 \land \text{Des-2}$

$\land (\lceil L \rceil \lor \lceil \neg L \rceil) \land (\lceil L \rceil \lor \lceil \neg L \rceil) \land (\lceil L \rceil \lor \lceil \neg L \rceil) \land (\lceil L \rceil \lor \lceil \neg L \rceil)$

For all state assertions $P$ and all real numbers $r_1, r_2 \in \mathbb{R}$.
Lemma 2.19

\[ \forall \ell \leq 29 \Rightarrow \int P = 0. \]

Claim:

\[ \forall \ell \leq 29 \Rightarrow \int P = 0. \]

Proof:

\[ \ell \leq 29 \Rightarrow \int P = 0. \]
\[
\begin{align*}
&\text{Proof:} \\
&\begin{cases}
0 = d \iff \emptyset = \emptyset \\
\tau = d \iff \{1\} = \{1\} \\
\tau = d \iff \{1\} = \{1\}
\end{cases}
\end{align*}
\]
A Calculus for DC

If time permits:
- Developing correctness may be difficult
- Different observables
- Intermediate design levels
- Requirements may be unrealizable

Obstacles (in a non-ideal world)
- Some laws of the DC integral operator
- Example: Gas Burner Controller

• Semantics-based Correctness Proofs

• Theorem 2.16
Des-1 and Des-2 is a correct design wrt. Req

• Lemma 2.19
Des-1 and Des-2 imply a simplified requirement Req-1

• Some Laws of the DC Integral Operator

• Lemma 2.17
Req-1 implies Req-2

• Obstacles (in a Non-Ideal World)

requirements may be unrealizable without considering plant assumptions

• intermediate design levels

• different observables

• proving correctness may be difficult

• If time permits:

A Calculus for DC

Content

Lemma 2.19

\[ (i) \] 
\[ | = \int P \leq \ell, \quad (iii) \] 
\[ \left| = \lceil \neg P \rceil \right| \]
\[ (ii) \] 
\[ | = \left( \int P = r_1 \right) \land \left( \int P = r_2 \right) \]
\[ (iv) \] 
\[ | = \lceil \rceil \leq \Rightarrow \int P = 0 \]

Claim:

\[ 0 = d f \leq \lceil \rceil = (0) \]
\[ \left( \neg a = d f \right) : \left( \neg a = d f \right) \leq (0) \]
\[ d^* = (0) \quad \rceil \neg a = d f \leq (0) \]
Methodology: The World is Not Ideal

(i) Choose a collection of observables \( \mathcal{O} \).

(ii) Provide specification \( \mathcal{R} \) (conjunction of DC formulae over \( \mathcal{O} \)).

(iii) Provide a description \( \mathcal{C} \) of the controller (DC formula over \( \mathcal{O} \)).

(iv) Prove \( \mathcal{C} \) correct with \( \mathcal{R} \) (prover = \( \mathcal{C} \) = \( \mathcal{R} \)).

(v) Prove \( \mathcal{C} \) complete over \( \mathcal{R} \).

That looks too simple to be practical.

Typical obstacles:

- It may be impossible to realize \( \mathcal{R} \).
- There are typically intermediate design levels between \( \mathcal{C} \) and \( \mathcal{R} \).
- \( \mathcal{C} \) and \( \mathcal{R} \) may use different observables.
- It doesn't consider properties of the plant.
- Proving validity of the implication is not trivial.
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 \( \text{Asm} \) on the input observables and verify correctness of \( \text{Ctrl} \) wrt. \( \text{Req} \) by proving validity of \( \text{Ctrl} \wedge \text{Asm} \Rightarrow \text{Req} \).

Any preference on the order (of (1) and (2))?

Shall we care whether \( \text{Asm} \) is satisfiable?

A top-down development approach may involve a level crossing for instance with a level crossing, we may assume that trains are not arbitrarily slow in the crossing. Otherwise we can’t make promises to the road traffic.

Intermediate Design Levels

A top-down development approach may involve:

- Key - specification/requirements
- Des - design
- Ctrl - implementation
- Req - specification/requirements

Then correctness is established by proving validity of

\( \text{Ctrl} \Rightarrow \text{Des} \)

and

\( \text{Des} \Rightarrow \text{Req} \)

We shall verify correctness of \( \text{Ctrl} \) wrt. \( \text{Req} \) by proving validity (from \( \text{Ctrl} \wedge \text{Asm} \Rightarrow \text{Req} \) and then concluding \( \text{Ctrl} \Rightarrow \text{Req} \) by transitivity).

Any preference on the order (of (1) and (2))?
• Assume, ' Req ' uses more abstract observables \( \text{Obs} A \) and ' Ctrl ' more concrete observables \( \text{Obs} C \).

• For instance:
  • in \( \text{Obs} A \): only consider one gas valve, open or closed — \( G : \{ 0, 1 \} \).
  • in \( \text{Obs} C \): may consider two valves and intermediate positions, for instance, to react to different heating requests — \( G_i : \{ 0, 1, 2 \} \).

• 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 \( \text{Link}_{C,A} \), the correctness of ' Ctrl ' wrt. ' Req ' amounts to proving:

\[
\begin{align*}
& \text{Ctrl} \land \text{Link}_{C,A} \Rightarrow \text{Req}.
\end{align*}
\]

• For instance, \( \text{Link}_{C,A} = \left\lceil (G_1 < 0 \land G_2 < 0) \lor (G_1 > 0) \right\rceil \).

---

Main options:

### Obstacle (iv): How to prove correctness?

• Sometimes a general theorem may fit (e.g. cycle times of PLC and its algorithms in Uppaal).

• Using proof rules from a calculus (later).

• By hand, on the basis of DC semantics (as demonstrated before).

• Main options:
  
  • Using proof rules from a calculus (later).

  • By hand, on the basis of DC semantics (as demonstrated before).

  • Sometimes a general theorem may fit (e.g. cycle times of PLC and its algorithms in Uppaal).

  • By hand, on the basis of DC semantics (as demonstrated before).

---

For instance:

• In \( \text{Obs} A \): only consider one gas valve, open or closed — \( G : \{ 0, 1 \} \).

• In \( \text{Obs} C \): may consider two valves and intermediate positions — \( G_i : \{ 0, 1, 2 \} \).

• Assume, Req uses more abstract observables than \( \text{Obs} A \) —\(^{iii)} \).
Design ideas for the behaviour of real-time system controllers can also be described using DC formulae. The correctness of a design idea with respect to requirements can principally be proven "on foot" using the DC semantics and analysis results.

This approach is not limited to over-simplified (gas) burner controllers. Therefore, DC semantics and analysis results can also be exploited by engineers in a "cookbook" manner, considering assumptions and intermediate designs in a step-by-step development.

Consider plant assumptions and different observables by invariants. Use intermediate designs in a step-by-step development and link different observables by invariants.

Consider other proof techniques. References for the behaviour of real-time system controllers can also be described using DC semantics.