

From Discrete Duration Calculus to Synchronous Observers

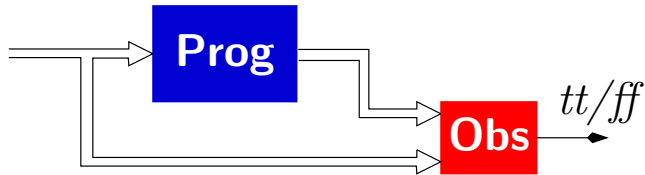
Laure Gonnord

Nicolas Halbwachs

Pascal Raymond

Vérimag, Grenoble

Specifying (safety) properties with synchronous observers



Well-known advantages:

- maximal power
- same language for programs and properties
- specifications are “executable”

Higher-level formalisms sometimes needed

Specification formalisms (e.g., temporal logics)

- simulation needed
- is decidability an issue?

Decision procedures for TL

$$\varphi \longrightarrow \mathcal{A}_\varphi$$

$$\sigma \in \mathcal{L}(\mathcal{A}_\varphi) \iff \sigma \models \varphi$$

$$\varphi \text{ satisfiable} \iff \mathcal{L}(\mathcal{A}_\varphi) \neq \emptyset$$

Easy if \mathcal{A}_φ rational

Model-checking of TL

$$P \models \varphi \iff \mathcal{L}(P \times \mathcal{A}_{\neg\varphi}) = \emptyset$$

Model-checking of TL

$$P \models \varphi \iff \mathcal{L}(P \times \mathcal{A}_{\neg\varphi}) = \emptyset$$

Rationality of \mathcal{A}_φ is of little practical interest, since P has generally an infinite number of states

Model-checking of TL

$$P \models \varphi \iff \mathcal{L}(P \times \mathcal{A}_{\neg\varphi}) = \emptyset$$

Rationality of \mathcal{A}_φ is of little practical interest, since P has generally an infinite number of states

Even in the finite-state case, the size of the automaton can be prohibitive.

Model-checking of TL


$$P \models \varphi \iff \mathcal{L}(P \times \mathcal{A}_{\neg\varphi}) = \emptyset$$

Rationality of \mathcal{A}_φ is of little practical interest, since P has generally an infinite number of states


Even in the finite-state case, the size of the automaton can be prohibitive.

→ symbolic automata, possibly extended with variables (counters)

Translating high level specification languages into symbolic automata (synchronous observers, safety properties)

$$\sigma \models \varphi \quad \text{iff} \quad \sigma \Rightarrow \Omega_{\varphi} \xrightarrow{\text{true}^*}$$
The diagram illustrates the translation of a high-level specification language into a symbolic automaton. It shows a state σ on the left, followed by the text "iff", and then a state σ on the left of a red box containing the symbol Ω_{φ} . An arrow points from σ to the red box, and another arrow points from the red box to the text true^* .

Translating high level specification languages into symbolic automata (synchronous observers, safety properties)

$$\sigma \models \varphi \quad \text{iff} \quad \sigma \Rightarrow \Omega_{\varphi} \xrightarrow{\text{true}^*}$$


- executable
- use in verification, runtime verification, testing...
- no explosion
- not necessarily finite state (counters...)

A well-known case: REGLO [Raymond96]

Translation of regular expressions into symbolic acceptors (Lustre observers)

Strictly linear size

A new experience: QDDC [Pandya 2000-03]

→ **one conclusion:** there are common (safety) properties which are surprisingly hard to express with observers, i.e., hard to check “on the fly”

QDDC: Quantified Discrete Duration Calculus

Examples

“Whenever p has been true continuously during at least n steps, q holds”

$$\square \left(\left(\prod p \wedge \eta \geq n \right) \Rightarrow \text{true} \curvearrowright [q]^0 \right)$$

or $p \xrightarrow{n} q$

“In any interval of duration d , p holds at least k times”

$$\square (\eta \geq d \Rightarrow \Sigma p \geq k)$$

QDDC: Semantics

States: sets of basic propositions (or Boolean valuations of propositional symbols) — models of propositions: $s \models P$

Traces: finite sequences of states

$$\sigma = \sigma_1 \sigma_2 \dots \sigma_n \quad |\sigma| = n$$

Windows: intervals in a trace

$$\sigma[b, e] = \sigma_b \sigma_{b+1} \dots \sigma_e \quad 1 \leq b \leq e \leq |\sigma|$$

Satisfaction of a formula:

by a window: $\sigma[b, e] \models \varphi$

by a trace: $\sigma \models \varphi \iff \sigma[1, n] \models \varphi$

QDDC: Semantics (cont.)

Examples:

$$\sigma[b, e] \models \eta \geq d \quad \text{iff} \quad (e - b) \geq d$$

$$\sigma[b, e] \models \Sigma p \geq k \quad \text{iff} \quad \text{Card}\{i = b..e \mid \sigma_i \models p\} \geq k$$

$$\sigma \models \square(\eta \geq d \Rightarrow \Sigma p \geq k)$$

$$\text{iff } \forall b, e, 1 \leq b \leq e \leq n,$$

$$(\sigma[b, e] \models \eta \geq d) \implies (\sigma[b, e] \models \Sigma p \geq k)$$

Our extensions: (forget decidability)

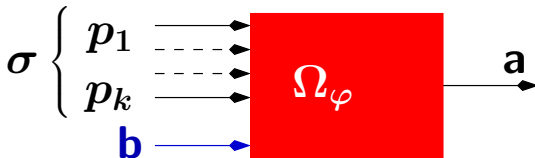
- Constants c (in η op c , ΣP op c , $P_1 \xrightarrow{c} P_2$) can be symbolic (parameters)
- The atomic propositions can be conditions on parameters (e.g., $c_1 \geq c_2$)

Observers of formulas (in Lustre)

inputs:

- σ (sequence of values for the propositions)
- a “starter” b (a Boolean, true only once)

output $a = \text{true}$ at t iff $\sigma, [t_b, t] \models \varphi$



Example 1: $\llbracket p \rrbracket$

True of each interval $[t_b, t]$ where p always holds

The observer output should

- be true before the (unique) occurrence of b
- take the value of p when b
- remain true as long as p is true

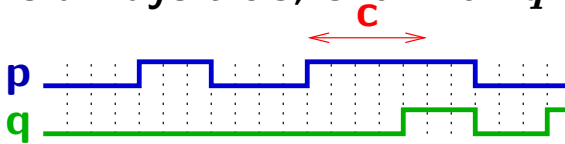
$$a = \text{before}(b) \text{ or } (\text{if } b \text{ then } p \text{ else } p \text{ and } \text{pre}(a))$$

where

$$\text{before}(b) = \text{not } b \rightarrow (\text{not } b \text{ and } \text{pre}(\text{before}(b)))$$

Example 2: $p \xrightarrow{c} q$

True on $[t_b, t]$ iff all subinterval of length c where p is always true, end with q true



$a = \text{before}(b)$ or

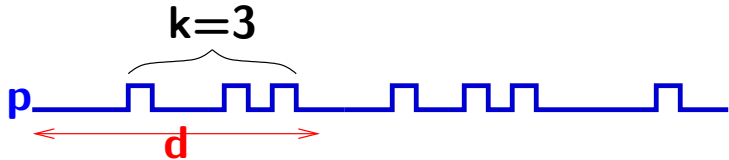
$$((\text{true} \rightarrow \text{pre}(a)) \text{ and } (\text{age}(p) \geq c \Rightarrow q))$$

where

$$\text{age}(p) = \text{if } p \text{ then } (0 \rightarrow \text{pre}(\text{age}(p))) + 1 \text{ else } 0$$

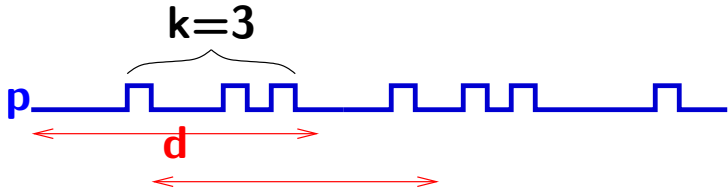
Example 3: $\square(\eta \geq d \Rightarrow \Sigma p \geq k)$

Each interval longer than d contains at least k occurrences of p



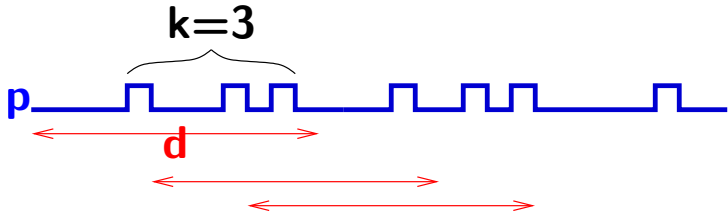
Example 3: $\square(\eta \geq d \Rightarrow \Sigma p \geq k)$

Each interval longer than d contains at least k occurrences of p



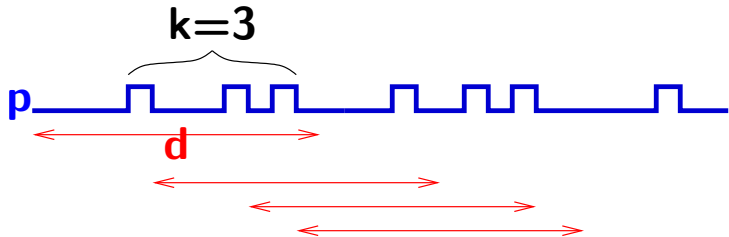
Example 3: $\square(\eta \geq d \Rightarrow \Sigma p \geq k)$

Each interval longer than d contains at least k occurrences of p



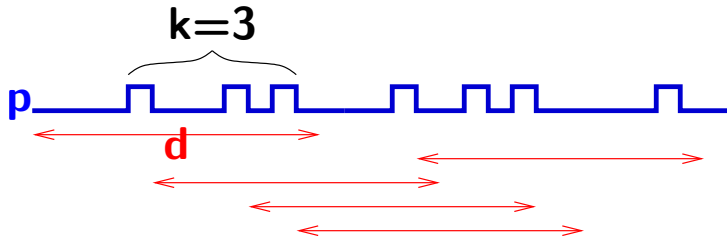
Example 3: $\square(\eta \geq d \Rightarrow \Sigma p \geq k)$

Each interval longer than d contains at least k occurrences of p



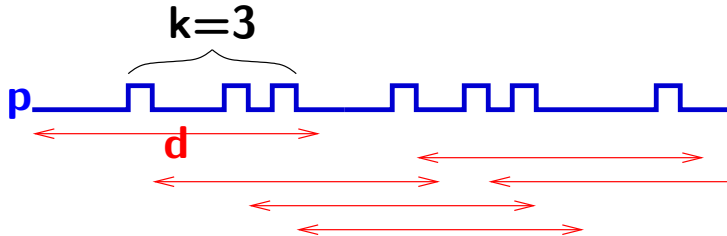
Example 3: $\square(\eta \geq d \Rightarrow \Sigma p \geq k)$

Each interval longer than d contains at least k occurrences of p



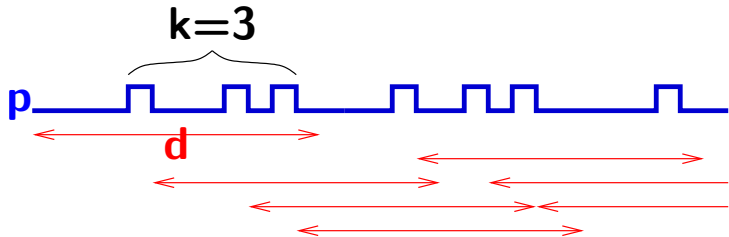
Example 3: $\square(\eta \geq d \Rightarrow \Sigma p \geq k)$

Each interval longer than d contains at least k occurrences of p



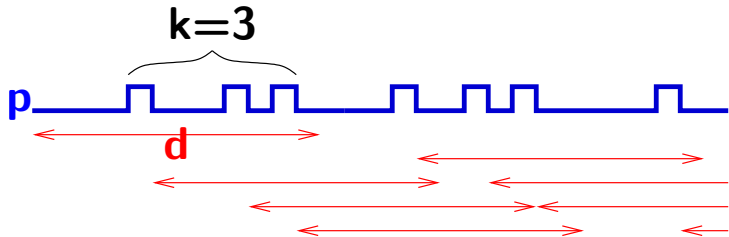
Example 3: $\square(\eta \geq d \Rightarrow \Sigma p \geq k)$

Each interval longer than d contains at least k occurrences of p



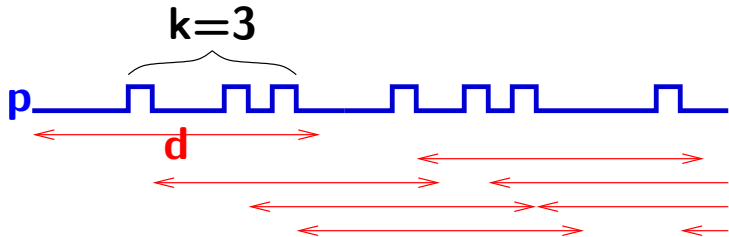
Example 3: $\square(\eta \geq d \Rightarrow \Sigma p \geq k)$

Each interval longer than d contains at least k occurrences of p



Example 3: $\square(\eta \geq d \Rightarrow \Sigma p \geq k)$

Each interval longer than d contains at least k occurrences of p



k (or d) counters needed!

Problem if d and k are parameters...

(for testing, one could use generic arrays. Out of reach of verification tools...)

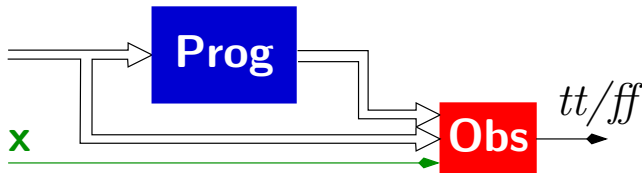
A solution: non-deterministic observers

Idea: non deterministically start a counter at each instant — by means of an additional input, say x (oracle).

A solution: non-deterministic observers

Idea: non deterministically start a counter at each instant — by means of an additional input, say x (oracle).

Since a model-checker verifies that the output is always true **whatever be the inputs**, it will verify whatever be x , so, for all intervals.



Example 3 (revisited): $\square(\eta \geq d \Rightarrow \Sigma p \geq k)$

a = before(b) or

((true \rightarrow pre(a)) and length < d or nb_p \geq k)

length = nb_since(true, x)

nb_p = nb_since(p, x)

where

nb_since(b1, b2) =

if before(b2) then 0

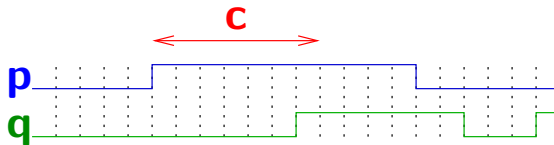
else if b2 then if b1 then 1 else 0

else pre(nb_since(b1, b2)) +

if b1 then 1 else 0

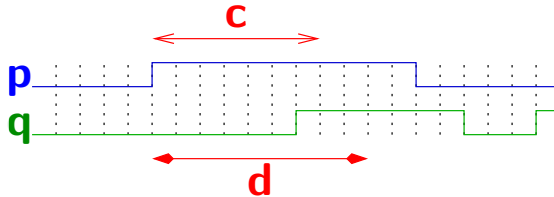
An example of verification

Prove that $(p \xrightarrow{c} q \wedge d \geq c) \implies (p \xrightarrow{d} q)$



An example of verification

Prove that $(p \xrightarrow{c} q \wedge d \geq c) \implies (p \xrightarrow{d} q)$



An example of verification (cont.)

Prove that, whatever be p , q , c , d **and** x , the output a is always true

$$\begin{aligned}
 a1 &= \text{before}(x) \text{ or } && \text{-- } p \text{ -}c\text{-} \rightarrow q \\
 &((\text{true} \rightarrow \text{pre}(a1)) \text{ and } (\text{nb_since}(p,x) < c \text{ or } q)) \\
 a2 &= \text{before}(x) \text{ or } && \text{-- } p \text{ -}d\text{-} \rightarrow q \\
 &((\text{true} \rightarrow \text{pre}(a2)) \text{ and } (\text{nb_since}(p,x) < d \text{ or } q)) \\
 \text{-- } (p \text{ -}c\text{-} \rightarrow q / d \geq c) &=> p \text{ -}d\text{-} \rightarrow q \\
 a &= (a1 \text{ and } d \geq c) => a2;
 \end{aligned}$$

An example of verification (cont.)

Prove that, whatever be p , q , c , d **and** x , the output a is always true

$$\begin{aligned}
 a1 &= \text{before}(x) \text{ or } && \text{-- } p \text{ -}c\text{-} \rightarrow q \\
 &((\text{true} \rightarrow \text{pre}(a1)) \text{ and } (\text{nb_since}(p,x) < c \text{ or } q)) \\
 a2 &= \text{before}(x) \text{ or } && \text{-- } p \text{ -}d\text{-} \rightarrow q \\
 &((\text{true} \rightarrow \text{pre}(a2)) \text{ and } (\text{nb_since}(p,x) < d \text{ or } q)) \\
 &\text{-- } (p \text{ -}c\text{-} \rightarrow q / d \geq c) \Rightarrow p \text{ -}d\text{-} \rightarrow q \\
 a &= (a1 \text{ and } d \geq c) \Rightarrow a2;
 \end{aligned}$$

Instantly proved, both by Lesar and Nbac

A fragment of QDDC recognizable by non-deterministic acceptors

Why only a fragment?

- Liveness properties expressible in QDDC
- Oracles can only be universally quantified
- Formulas $\exists p\varphi$ and $\varphi_1 \frown \varphi_2$ need existentially quantified oracles

3-levels syntax

$$\varphi ::= \neg\psi$$

$$\psi ::= \xi \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2 \mid \exists p \psi \mid \psi_1 \rightsquigarrow \psi_2$$

$$\xi ::= [P]^0 \mid \llbracket P \rrbracket \mid \eta \text{ op } c \mid \Sigma P \text{ op } c \mid \neg\xi$$

Example: $\Box(\eta > c \Rightarrow \Sigma p \geq d)$

Translation into basic QDDC:

$$\neg (true \curvearrowright (\eta > c \wedge \Sigma p < d) \curvearrowright true)$$

Translation into Lustre:

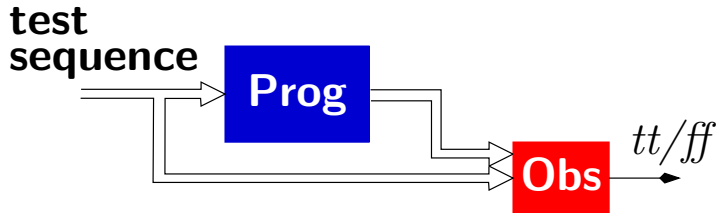
$$a = \text{not} (\quad \text{nb_since}(\text{true}, x) > c \text{ and} \\ \quad \quad \quad \text{nb_since}(p, x) < d)$$

Ok for verification (since the verification tools universally quantify over oracles)

Not suitable for property simulation, and not very good for testing (a test may succeed for some values of the oracles, but fail for others)

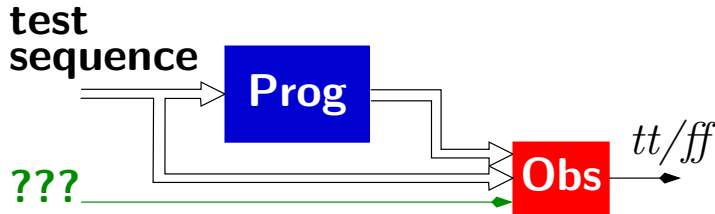
Ok for verification (since the verification tools universally quantify over oracles)

Not suitable for property simulation, and not very good for testing (a test may succeed for some values of the oracles, but fail for others)



Ok for verification (since the verification tools universally quantify over oracles)

Not suitable for property simulation, and not very good for testing (a test may succeed for some values of the oracles, but fail for others)



→ a deterministic fragment

Conclusion

- **There are common safety properties which are not obvious to translate into symbolic acceptors**
- **Non-deterministic acceptors increase the descriptive power**

Conclusion (cont.)

- **What has been done:**
 - Identification of a useful fragment of QDDC which can be translated into non-deterministic acceptors
 - Identification of a more restrictive deterministic fragment
 - Translations into Lustre (and proof of correctness)

Further works...

- What about really complex formalisms (SUGAR...)?