Formalizing Timing Diagram Requirements:

in Discrete Duration Calculus

Rajmohan Mattaplacket[1], Paritosh Pandya[1], Amol Wakankar[2]

Tata Institute of Fundamental Research, Mumbai
Bhabha Atomic Research Center, Mumbai

November 25, 2017

Formal Requirement Modelling and Analysis

- Domain: Behavioural requirequirements over Embedded system controllers/Digital Hardware modules
- Formal specification notation: some form of temporal logic
- In practice: Informal specification, Heterogenous, Visual (UML timing diagrams, state machines, MSC), Structured text.

Promise of Formal Specification

- Unambiguous
- Requirement Analysis: Consistency, Model Visualization, Completeness, Implication checking, Realizability checking.
- Verification: Model checking, Runtime verification, Automatic test suite generation.
- Synthesis: Automatic construction of controller which matches the specification.

In this paper: A logic for formalizing timing diagram requirements

Formalizing Timing Diagram Properties

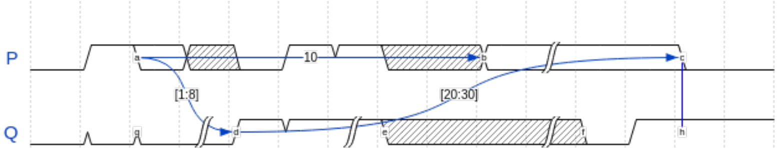
Fisler 2005

The less than satisfactory adoption of formal methods in timing diagram domain can be partly attributed to the gulf that exists between graphical timing diagrams and textual temporal logic – expressing various timing dependencies that can exist among signals that can be illustrated so naturally in timing diagrams is rather tedious in temporal logics.

As a result, hardware designers use timing diagrams informally without any well defined semantics which make them unamenable to automatic design verification techniques.

Outline

- Timing diagrams
- From Timing diagrams to Temporal Logic
- SECENL $\subset$ QDDC
- Encoding Timing diagrams in SECENL, and comparsion
- Elementary complexity of SECENL automaton construction. Tool DCTOOLS.



Components of Timing Diagrams

- Timing diagram $T = (W, \Sigma, C, \Theta)$
- Σ set of propositions
- $W(p)$ gives the waveform of $p \in \Sigma$
- Θ a set of named positions in waveforms.
- C set of constraints between pairs of named positions.
  - Precedence constraints
  - Timing constraints

waveform W_p - 01a : 2x011xb : x2|220c : 00
waveform W_q - 00a : 0|d : 11|e : xxx|f : 01c : 11
timing constraints: d-a∈[1:8], c-d∈[20:30], b-a∈[10:10]

Timing Diagram Textual Syntax

Timing diagram $T = (W, \Sigma, C, \Theta)$

Waveform syntax

$$\pi := 0 \;\|\; 1 \;\|\; 2 \;\|\; 0| \;\|\; 1| \;\|\; 2| \;\|\; x| \;\|\; u : \pi \;\|\; \pi_1 \pi_2,$$
$$\textit{where} \quad u \in \Theta$$

0 denotes *low*, 2 *any*, and "|" the *stuttering* operator. Thus, 1| gives arbitrarily long high signal, and x|, arbitrarily long unchange.

Constraints

- C is a list of constraints
- Example constraint: $(c, d, [20 : 30])$
- Set of Constraints $\Theta \times \Theta \times \mathit{Intv}(\mathbb{N})$

- Syntax is adapted from Wavedrom2 which draws the picture.
- DCTOOLS translates TD to logic SECENL.

Timing Diagram Logics

- LTL [Dill, Emerson,1997] and CTL
- Extensions: Timing Diagram Logic [Fisler,1999], Pipeline operator [Chockler, Fisler,2005], SRTD [Amla,Emerson,Kurshan,Namjoshi,2000]
- PSL/Sugar (IEEE1850 [2005]): formulated initially by Accelera Consortium
- Interval Temporal Logic [Moszkowski, 1983]
  Duration Calculus [Zhou, Hoare, Ravn, 1990]
- In this paper, SECENL a subset of QDDC [P.,1996] and CTL*(DC) [P., 2001]

Discrete-time Duration Calculus

QDDC logic of finite (non-empty) state seqeunces.

```
req   1  0  1  1  0
ack   0  0  0  0  1
```

We define $\sigma, [b, e] \models D$.

Example `<req> ^ [!ack] ^ <ack>`

- Interval temporal logic
- Quantitative Measurements of Time

Example In any interval of 20 or more cycles where request is continuously high there must be at least 3 *ack* signals.

```
[]( [[req]] && slen >= 20  => scount ack >= 3)
```

QDDC Syntax

Let P $\in$ *Prop*(Σ), $c \in \mathbb{N}$, D1,D2 $\in$ *QDDC*. Let $\in$ { <=, <, =, >, >=} Then syntax of QDDC:

```
<P> | [[P]] | slen ~ c | scount P ~ c |
D1^D2 | D* | D1 && D2 | !D |
(exists P. D)
```

QDDC: Syntax and Semantics

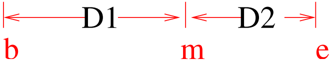
$\sigma, [b, e] \models$ <P> **iff** $b = e$ and $\sigma, b \models P$

$\sigma, [b, e] \models$ [[P]] **iff** for all $t : \ b \leq t \leq e. \ \ \sigma, t \models P$

$\sigma, [b, e] \models$ [P] **iff** $b < e$ and for all $t : \ b \leq t < e. \ \ \sigma, t \models P$

$\sigma, [b, e] \models$ D1^D2 **iff** for some $m : b \leq m \leq e.$
$\sigma, [b, m] \models$ D1 and $\sigma, [m, e] \models$ D2

If for some m



Example: [P]^[!P]^[[P]]

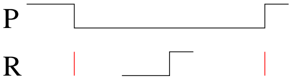
A valid formula: <P> <=> <P>^<P>^<P>

Derived Operators

- For some subinterval D: <>D $\stackrel{\text{def}}{=}$ true^D^true
- For all subintervals D: []D $\stackrel{\text{def}}{=}$! <> !D

Validity in execution $\sigma \models D \quad \textbf{iff} \quad \sigma, [0, \#\sigma - 1] \models D$

Example: [](<down(P)>^[!P]^<up(P)> => !<>([!R]^[R])



Measurement Formulae

Measurement Terms $\quad$ `slen` | `scount P`

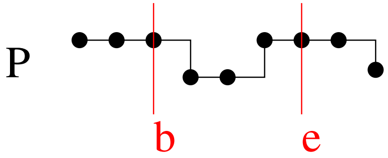
$$eval(\texttt{slen})(\sigma, [b, e]) \stackrel{\text{def}}{=} e - b$$

$$eval(\texttt{scount}\ \ \texttt{P})(\sigma, [b, e]) \stackrel{\text{def}}{=} \sum_{i=b}^{e} \left\{ \begin{array}{ll} 1 & if \ \ \sigma, i \models P \\ 0 & otherwise \end{array} \right\}$$

Measurement Formula *mt op c*

where $op \ \in \ < \ | \ > \ | \ = \ | \ \leq \ | \ \geq$.

Measuring Counts and Durations



$eval$(`slen`) = 4

$eval$(`scount P`) = 3

Examples

```
[]( [[req]] && slen >= 20  => scount ack >= 3)
```

Between any two *P* phases there are at least 300 cycles.

```
[] ( < down(P)>^[!P]^<up(P)>  =>  (slen >= 300) )
```

- Minimum Separation
- Upper bound
- Persistence
- Arrow operators [Ravn94]

Quantification `exists p: D`

$\sigma, [b, e] \models$ (`exists p: D`) **iff** $\sigma', [b, e] \models D$ for some p-variant σ'

Formula Automaton Construction

Theorem (Automata Theoretic Decidability of *QDDC*)

- *For each* $D \in QDDC$ *we can effectively construct finite state automaton* A_D *such that* $L(D) = L(A_D)$.
- *For each FSM* A *we can effectively construct* $D_A \in QDDC$ *such that* $L(A) = L(D_A)$.

Tool DCVALID – next slide.

Problem

Size of minimum automaton can be non-elementary in size of formula in the worst case. Thus formula of size n can give minimal automaton of size $O(2^{2^{2^{\cdots}}})$, tower of height n.

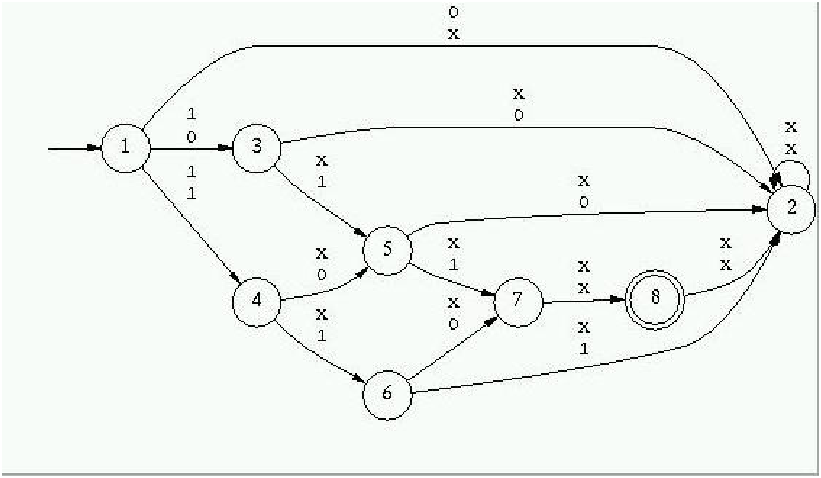
DCVALID: Validity/Model Checker for QDDC formulas

- Constructs deterministic finite state automaton $\mathcal{A}(\mathcal{D})$ for QDDC formula D.
- The automaton is used as a synchronous observer to model check QDDC properties of Esterel, SMV, Verilog, SCADE/Lustre and SAL models.
- Uses efficient BDD-based representation of automata using MONA.
- Constructs automaton for formula bottom up keeping each automaton in minimal deteterminstic form.

[RTTOOLS2001, TACAS2001, SLAP2002, AVOCS2004, FSTTCS2005, TACAS2006]

DCVALID Example

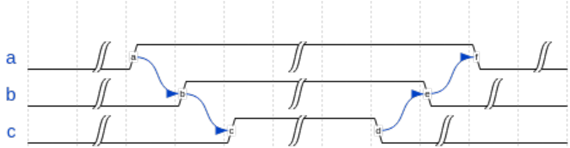
$$(\langle P \rangle \frown true) \wedge (slen = 4) \wedge (sdurQ = 2)$$



SECE: Semi Extended Chop Expressions

- SECE is QDDC without negation and quantification operators.

```
<P> | [[P]] | slen op c | scount P op c |
D1^D2 | D* | D1 && D2
```



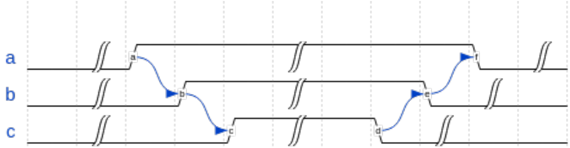
- SECE adequately captures a collection of waveforms (i.e. W)

```
[!a]^([a])^[[!a]] && [!b]^([b])^[[!b]] &&
[!c]^([c])^[[!c]]
```

Nominals: Capturing Waveform Constraints

- Nominals are propositions which uniquely mark specific positions in word.
- Used for synchronization between formulae.
- Let D be a SECE formula over $\Sigma \cup \Theta$.
- `(ex1 u: D)` where $D \in SECE$ and $u \in \Theta$ is called SECEN formulas.
- `(ex1 u: D)` = `(exists u: scount(u)=1 && D)`
  `(all1 u: D)` = `(all u: (scount(u)=1 => D))`

Timing Diagrams to SECEN



SECEN Formula

```
ex1 ua,ub,uc,va,vb,vc:
-- waveforms
    [!a]^<ua>^[a]^<va>^[[!a]] &&
    [!b]^<ub>^[b]^<vb>^[[!b]] &&
    [!c]^<uc>^[c]^<vc>^[[!c]] &&
-- constraints
    true^<ua>^slen>0^<ub>^true &&
    true^<ub>^slen>0^<uc>^true &&
    true^<vc>^slen>0^<vb>^true &&
    true^<vb>^slen>0^<va>^true
```

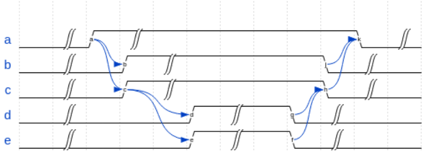
Size $O(n)$

Equivalent Formula without Nominals

```
[!a && !b && !c] ^ [a && !b && !c] ^ [a && b && !c] ^
[a && b && c] ^
[!a && !b && !c] ^ [a && !b && !c] ^ [a && b && !c]
```

Size $O(n^2)$.

Example 2

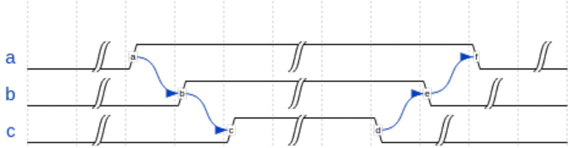


SECEN Formula

```
ex1 ua,ub,uc,va,vb,vc:
  [!a]^<ua>^[a]^<va>^[[!a]] &&
  [!b]^<ub>^[b]^<vb>^[[!b]] &&  ...
-- constraints
  true^<ua>^slen>0^<ub>^true &&  true^<ua>^slen>0^<uc>^true &&
  true^<uc>^slen>0^<ud>^true &&  true^<uc>^slen>0^<ue>^true &&
  true^<vd>^slen>0^<vc>^true &&  true^<ve>^slen>0^<vc>^true &&
  true^<vc>^slen>0^<va>^true &&  true^<vb>^slen>0^<va>^true
```

Writing this without nominals is tricky!

Unordered Stack



SECEN Formula

```
ex1 ua,ub,uc,va,vb,vc:
    [!a]^<ua>^[a]^<va>^[[!a]] && ..
    [!c]^<uc>^[c]^<vc>^[[!c]] &&
-- constraints
    ext^<u1>^ext^<u2>^ext^<u3>^ext^
        <v3>^ext^<v2>^ext^<v1>^ext &&
    Bijection(ua,ub,uc,va,vb,vc,u1,u2,u3,v1,v2,v3)
```

Size $O(n^2)$

Stating this without nominals requires disjunction over all possible stack orders. Size $O(n!)$

SECEN with Nominals

Main Features

- Natural and compositional translation of timing diagrams into *SECEN*
- Exponential succinctness as compared with *SECE* (and *SERE* of PSL)
- Elementary automaton construction.

Theorem

For every $D \in SECEN$ of size n, we can construct $A(D)$ of size $2^{2^{2^n}}$ such that $L(D) = L(A(D))$.

SECENL: Specifying Limited Liveness

- Modalities of occurence of patterns of behaviour
  Inspired by LSC of UML 2.0.
- Making good things happen – within known bounds.

Syntax of SECENL formula ϕ

Let D_i be SECEN formulas. Let Θ be a set of nominals.

$$
\begin{array}{l}
\textbf{pref}(D) \\
\textbf{anti}(D) \\
\textbf{init}\forall^1\Theta : (D_1/D_2) \\
\textbf{implies}\forall^1\Theta : (D_1 \rightsquigarrow D_2) \\
\textbf{follows}\forall^1\Theta : (D_1 \rightsquigarrow D_2/D3) \\
\textbf{triggers}\forall^1\Theta : (D_1 \rightsquigarrow D_2/D3) \\
\phi_1 \wedge \phi_2 \;\mid\; \neg\phi
\end{array}
$$

A SECENL formula ϕ is equivalent to a QDDC formula (using negations) called $\aleph(\phi)$.

Reduction to QDDC

- **anti**(D) – pattern D must not occur anywhere in behavior.

$$\aleph(\textbf{anti}(D)) \quad = \quad$$ `[](!D)`

- **implies**$\forall^1\Theta : (D_1 \rightsquigarrow D_2)$

$$\aleph(\textbf{implies}\forall^1\Theta : (D_1 \rightsquigarrow D_2)) =$$

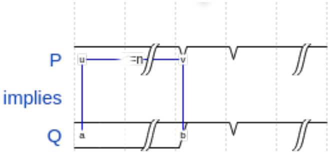
```
[]( all1 Theta: (D1 => D2))
```

- **follows**$\forall^1\Theta : (D_1 \rightsquigarrow D_2/D3)$

$$\aleph(\textbf{follows}\forall^1\Theta : (D_1 \rightsquigarrow D_2/3)) =$$

```
[]( all1 Theta,u: (D1^<u>^D3 => true^<u>^D2^true))
```

Example: Lags(P,Q,n)



SECENL formula:

implies$\forall^1 u$:
(([P] && slen=n)^<u>^[[P]]) $\rightsquigarrow$
true^<u>^[[Q]])

Minepump Example:

```
lags(HCH4,Alarm,3) && lags(HH2O,Alarm,3) &&
lags((!HCH4 && !HH2O),!ALARM,3)
```

Other Properties

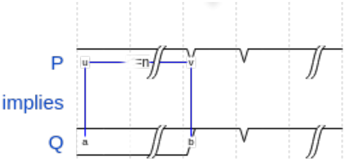


Figure : lags(P,Q,n).

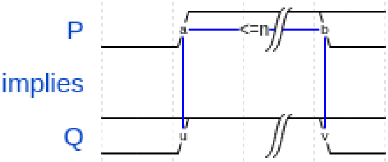


Figure : track(P,Q,n)

Figure : sepration(P,n)

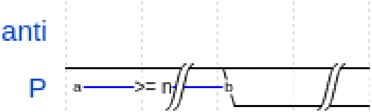


Figure : upperbound(P,n)

Formula automaton construction

Theorem (Elementary Automaton Construction)

For every $D \in SECENL$ of size n, we can construct $A(D)$ of size $2^{2^{2^{2^{2^n}}}}$ (tower of height 5) such that $L(D) = L(A(D))$.

In practice not so bad! Compare this with PSL with SERE which gives tower of height 4.

DCTOOLS: Architecture

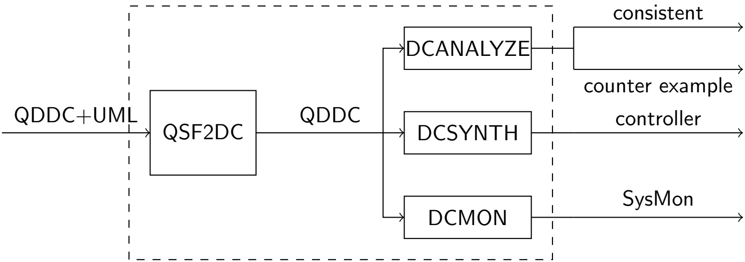


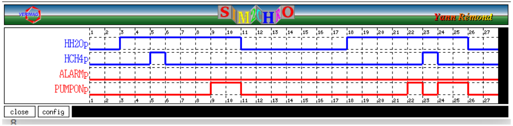
Figure : DCTOOLS.

Demonstrators

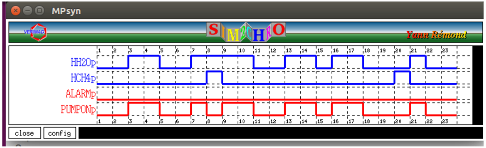
- Minepump Controller Specification (proceedings)
  Automatic Controller Synthesis
- Synchronous Bus Arbiter with diverse latency properties
  Automatic Controller Synthesis, Automatic Monitor Synthesis
- Alarm annunciation System for a plant
- Discordance logic for a plant
- Complete AMBA Bus AHB arbiter specification (in progress)
- Specification of self navigating and parking robot car controller
  Automatic controller synthesis (in progress)

Synthesis with Soft Goals

Soft requirement: `(!$PUMPON$) >> (!$ALARM$)`



Soft Requirement: `$PUMPON$ >>(!$ALARM$)`

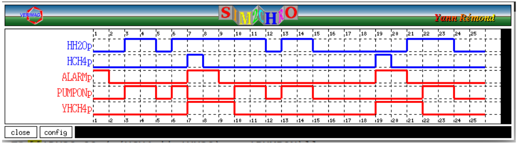


Synthesis with Soft Goals

Soft Requirement:

((!$YHCH4p$)|(!$PUMPONp$))>>($PUMPONp$)

where YHCH4p : true^(slen=2 && <><HCH4p>)



DCSynth Performance

Problem	Lily		Acacia+		DCSynth	
	time (Sec)	Memory /States	time (Sec)	Memory /States	time (Sec)	Memory /States
$Arb^{hard}(4,4)$	161.9	172.6/ 108	0.4	29.8/ 55	0.09	5.0/ 50
$Arb^{hard}(5,5)$	TO[a]	-	11.4	71.9/ 293	4.8	33.4/ 432
$Arb^{hard}(6,6)$	-	-	TO	-	80	1053.0/ 4802
$Arb^{hard}(7,7)$	-	-	TO	-	-	MO[a]
$Arb^{tok}(8)$	TO	-	46.44	77.9/ 73	1.9	12.8/ 8
$Arb^{tok}(10)$	TO	-	NC[a]	-	137	53.0/10
$Arb^{tok}(12)$	TO	-	NC	-	TO	255.0/ 12
MinePump	TO	-	NC	-	0.06	50/ 32

[a] TO=timeout, MO=memory out and NC=synthesis inconclusive.

Combining SECENL and LTL

- Let $\phi \in$ SECENL. Then LTL[SECENL] contains LTL formulae where SECENL formulae are used as propositional letters.
- Semantics: Given finite or infinite behaviour ρ and $i \in dom(\rho)$

$$\rho, i \models \phi \ \ \textbf{iff} \ \ \rho, [0, i] \models \phi$$

- Easy to model check!
- Tool ctldc modelchecks LTL[SECENL] specification against LUSTRE or SMV model by using underlying model checker (see [P. TACAS2001]).
- All the standard LTL synthesis algorithms can be extended to LTL[SECENL].

Conclusions

- A logic for requirements is a pragmetic choice between expressiveness and decision complexity.
- SECENL is a proposal which differs from state-of-the-art-logics like PSL or Regular LTL.
  - Semi Extended Chop Expressions with intersection and counting
  - Nominals for synchronization
  - Limited Liveness Properties: Interval logic connectives and implication without nesting.
- LTL[SECENL] is a simpler combination of LTL and SECENL. Easy to model check or synthesize.
- Tool support with DCTOOLS.