Formal Description & Automated Verification

Embedded Systems
Formal Methods
Lecture 5

Farn Wang
Dept. of Electrical Engineering
National Taiwan University

1

Contents

- Timed Automata
- Model-checking of timed automata
- Symbolic algorithm for the model-checking of timed automata
- Linear hybrid automata and symbolic verification procedure

2

Real-time system modeling

What is a Real-Time System？

The correctness of a system is defined with both the output values and the time the output appears

- avionics
- battlefield management
- missile guidance
- submarine sonar
- nuclear plant
- life monitor
- chemical reactor
- flight control

3

Real-time system verification

Correctness, non-real-time systems

- **For finite computations, defined with relations between precondition and postcondition**
- **For infinite computations, defined with the ordering of events.**
  - Buechi automata
  - mutual exclusion
  - fairness

4

Real-Time system modeling

What is a Real-Time System？

The correctness of a system is defined with both the output values and the time the output appears

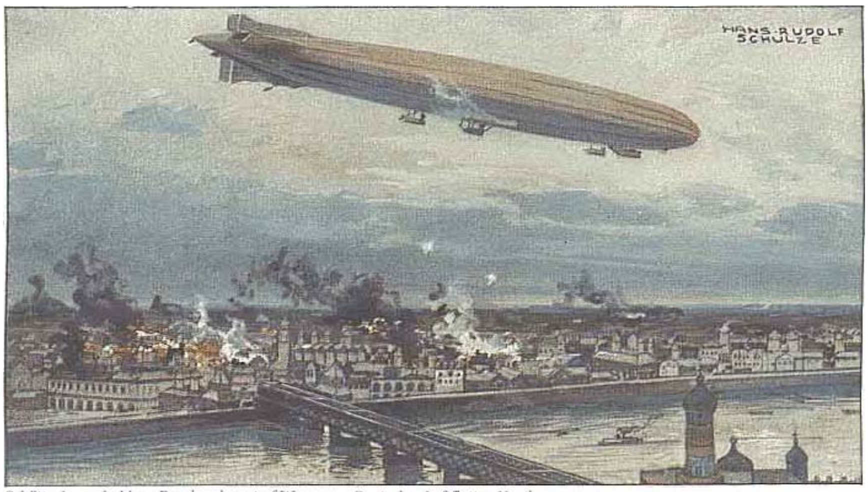
Example:

synchronous machine guns

- **an untimely output may be worse than no output.**

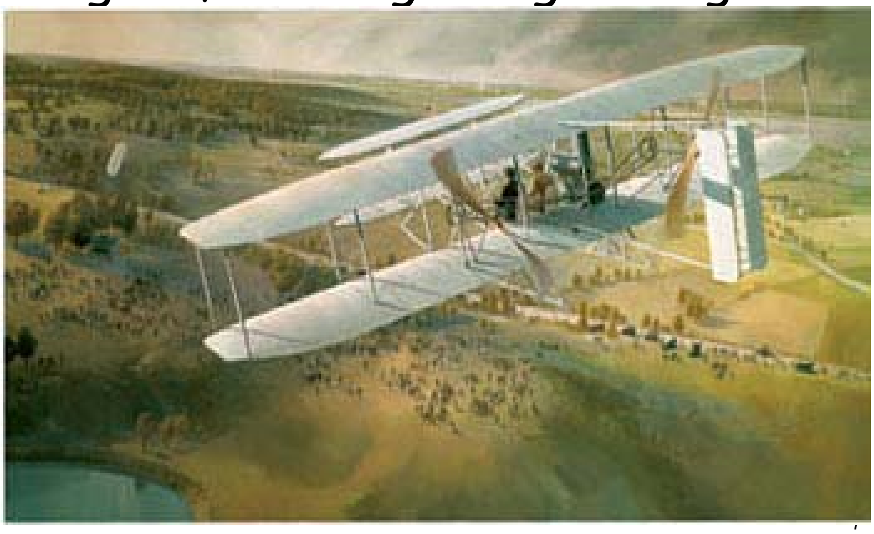
5

Schütte-Lanz SL2 airship bombing Warsaw



Schütte-Lanz airship – *Bombardment of Warsaw* – Deutscher Luftflotten Verein

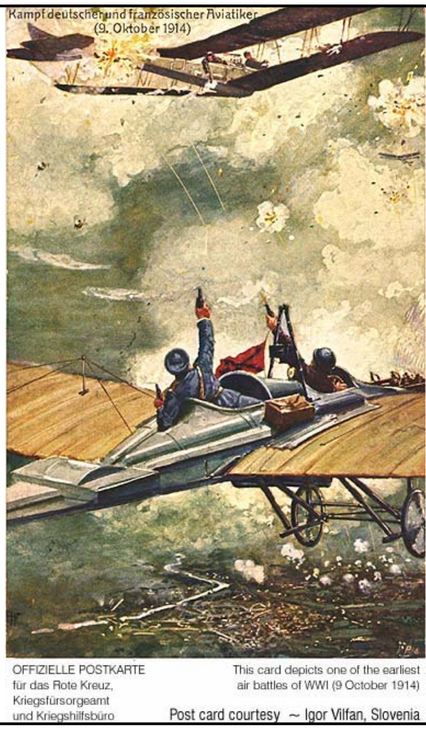
Wrights, the beginning of flight era



The beginning of air-combat

An imaginary solution to air-combat

Difficult to aim in 3D maneuver



8

A solution to air-combat

9

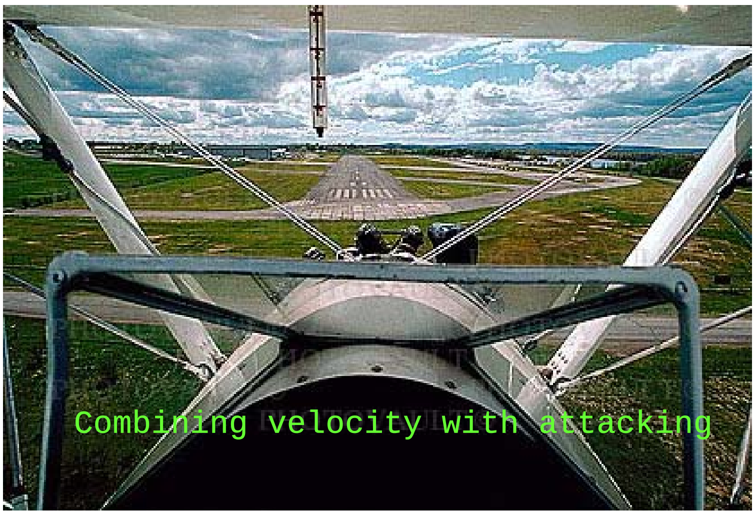
Another solution to air-combat

)

Aircombat (Dogfight) in WWI

11

A nice idea to air-combat



12

A nice idea to air-combat (cont'd)

13

A problem with the idea

If it is not real-time, …

14

A solution to air-combat

Synchronous machine gun

15

Sopwith Camel

Sopwith Camel, Old Rhinebeck Aerodrome. Photo by Nels Anderson

16

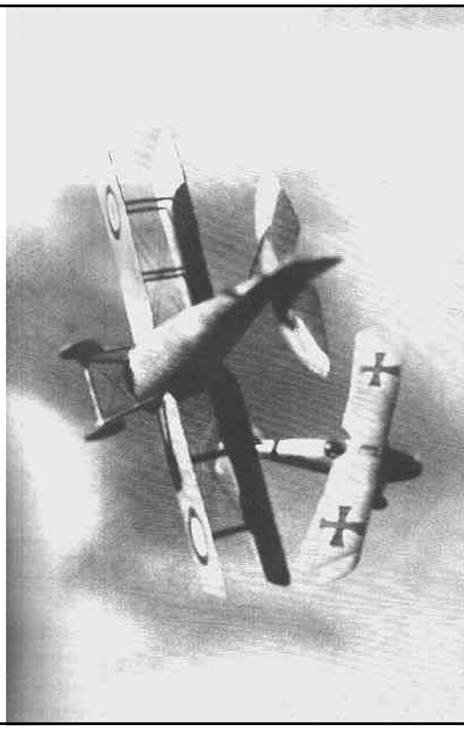
SPAD
VII
S.254

17

SPAD VII S.254

18

WWI dogfight

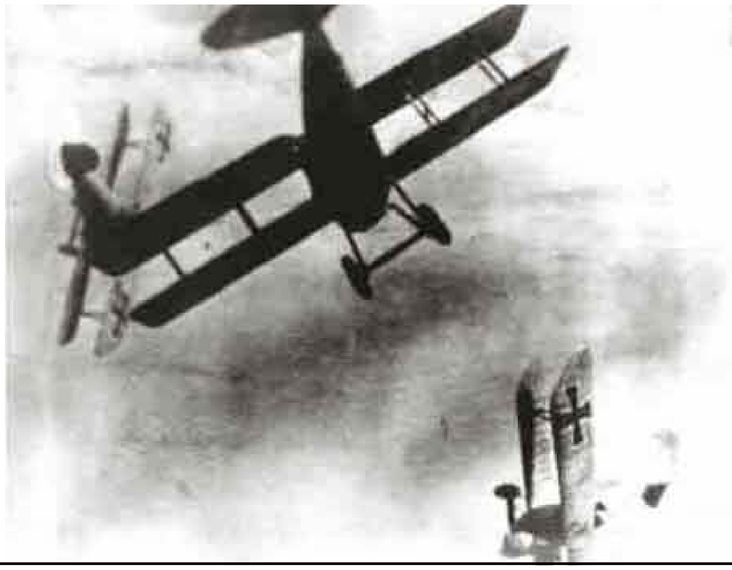


19

WWI dogfight

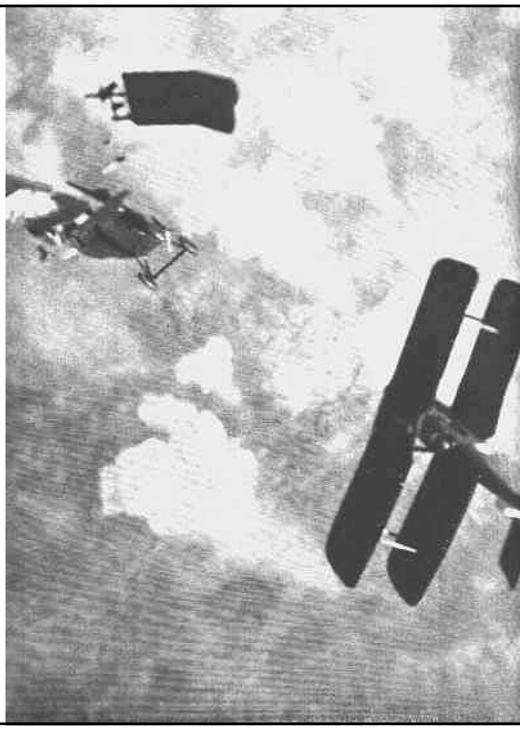
20

WWI dogfight

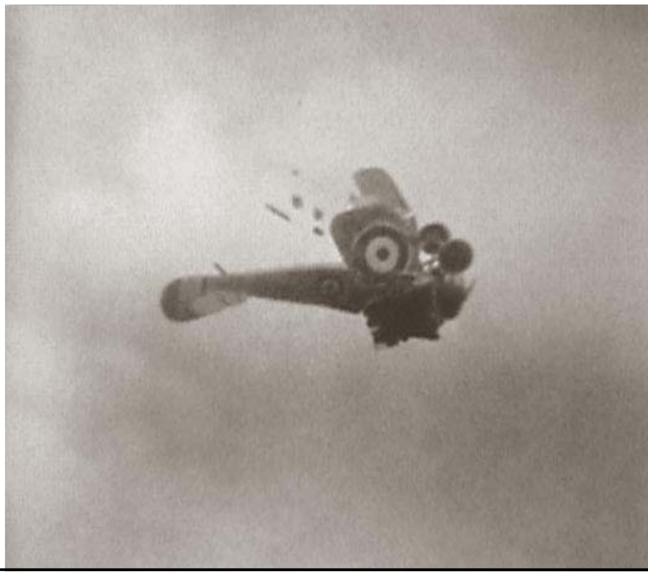


21

Disintegration in dogfight

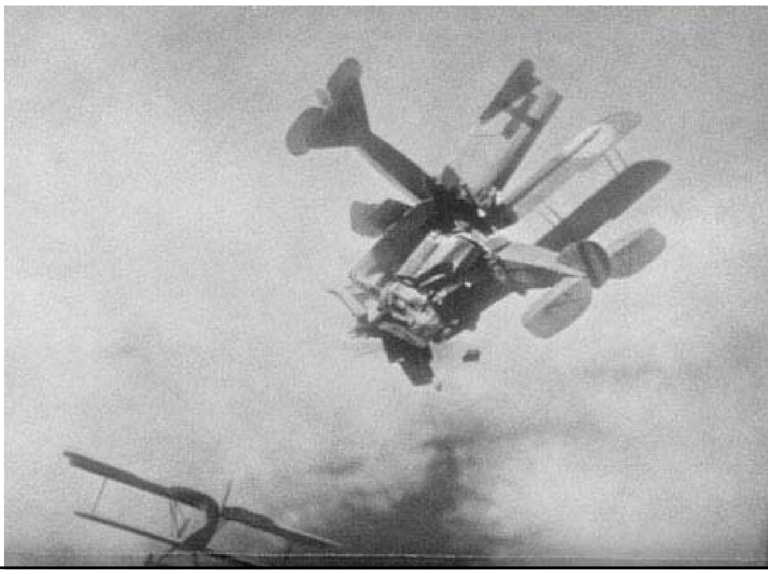


Disintegration in dogfight



23

Collision in dogfight



24

Pilot falling from burning Albatros

25

Why dense-time clock？

Why not discrete clock ?

- discrete clocks can be readily modeled with finite-state automata.
- In engineering, as long as the clock resolution is fine enough, we can have enough timing precision of the models.
- In fact, all clock readings in a digital computer must be recorded as bits.

26

Why dense-time clock ?
- the arguments (1/2)

For model timing precision,

- **distributed clocks – discrete clocks may not increment their readings at the same time.**
  - To check the ordering of clock ticks at many sites engender the same model complexity as dense-time models.
- **How fine is fine enough ?**
  - In theory, there is never a fine enough resolution that guarantees the verification correctness.
  - For safety-critical systems, dense-time models could be a necessity.

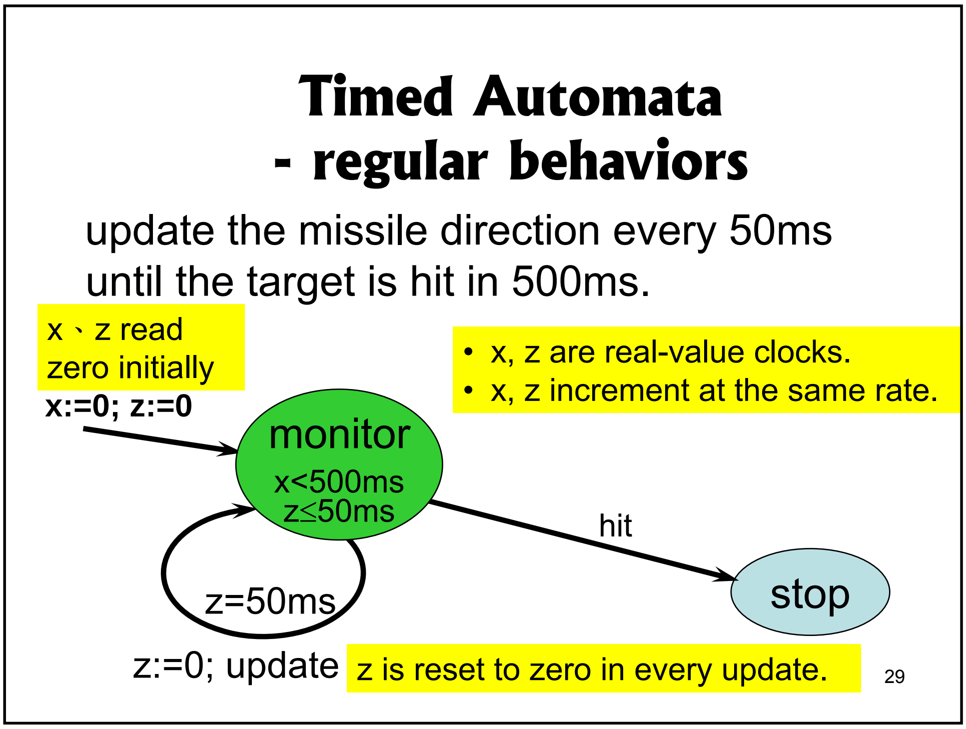
27

Why dense-time clock ?
- the arguments (2/2)

For verification complexity

- fine time resolution
  → large timing constants
  → high verification complexity
- dense linear constraints → P
  discrete linear constraints → NP-complete

28


Timed Automata
- regular behaviors
update the missile direction every 50ms
until the target is hit in 500ms.
x、z read
zero initially
x:=0; z:=0
• x, z are real-value clocks.
• x, z increment at the same rate.
monitor
x<500ms
z≤50ms
hit
stop
z=50ms
z:=0; update
z is reset to zero in every update.
29


Time
Au
a
ors
• clock readi
ls.
• In between
ck's
rea
• All clo
s at the
same
Sin
e,
• the sta
size and
• state
• No
next states.'
The basic challenge
in verification
How can we check
an infinite state-space
with finite amount
of time and space ?
30

Timed automata - syntax

Be sure how to read BNF !

- used for define syntax of context-free language
- important for the definition of
  - automata predicates and
  - temporal logics
- Used throughout the lectures!
- In exam: violate the syntax rules ➔ **no credit.**

A ::= (M) | A1 + A2 | A1 – A2
M ::= (A) | M1 * M2 | M1 / M2

31

Timed automata - syntax

State predicates :

Given a proposition set P and a clock set X

$$\eta ::= p \mid x \sim c \mid \neg\eta_1 \mid \eta_1 \vee \eta_2$$

BNF

$$p \in P;\ x \in X;\ c \in N;\ `\sim' \in \{\leq, \geq, <, >, =\}$$

Example:

male ∧ single ∧ age ≤ 30
lonely ∧ male ∧ single ∧ today+2≤nextMonday
monitor ∧ x+5 < ReportPeriod

32

Timed automata - syntax

State predicates :

Given a proposition set P and a clock set X

$$\eta ::= p \mid x \sim c \mid \neg\eta_1 \mid \eta_1 \vee \eta_2$$

$$p \in P;\ x \in X;\ c \in N;\ `\sim' \in \{\leq, \geq, <, >, =\}$$

Shorthands:

true	$\equiv$	$x = x$
$\eta_1 \wedge \eta_2$	$\equiv$	$\neg((\neg\eta_1)\vee(\neg\eta_2))$
$\eta_1 \rightarrow \eta_2$	$\equiv$	$(\neg\eta_1)\vee\eta_2$

B(*P*, *X*): **All state-predicates of *P and X***

33

Timed automata - syntax

$A=\langle Q, I_0, \mu, E, \tau, \pi\rangle$

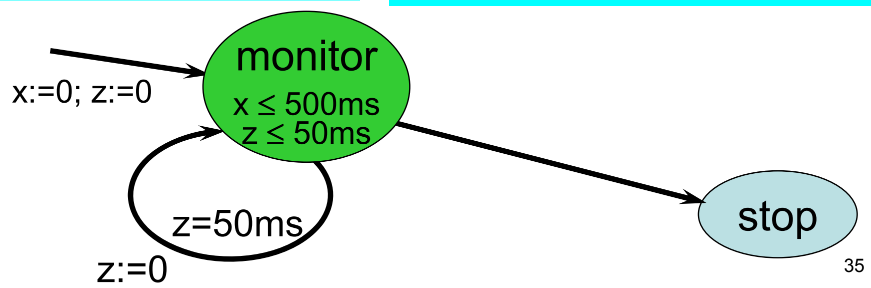
- **Q,** ***a finite set of control locations***
- **$I_0 \in$ B(P,X),** **an initial condition**
- **$\mu : Q \rightarrow$ B(*P*,*X*),** **invariance conditions at the locations**
- **$E \subseteq Q \times Q$,** **a set of transitions**
- **$\tau : E \rightarrow$ B(*P*, *X*),** **triggering conditions of the transitions**
- **$\pi : E \rightarrow 2^X$,** **sets of clocks to reset at transitions**

34

Timed automata - syntax

A=⟨*Q*, I_0, μ, *E*, τ, π⟩
Q ={monitor, stop}
I_0 = monitor∧x=0 ∧z=0
μ(monitor) = *x* ≤ 500∧*z* ≤ 50
μ(stop)= *true*

E={(monitor,monitor),(monitor, stop)}
τ(monitor, monitor)= *z* =50
τ(monitor, stop)= *true*
π(monitor, monitor)= {*z*}
π(monitor, stop)= { }



35

Timed automata - semantics

clock valuatoin

defines the values of all Boolean variables.

defines the non-negative real values of all clocks

A=⟨Q, I_0, μ, E, τ, π⟩

states: (*q*, ν)

- ***q*** ∈ **Q**
- ν : ***P*** → **{*true*,*false*}** ∪ ***X*** → ***R***$^+$

★ν ⊨ μ(*q*), **clock values must satisfy invariance condition.**

36

Timed automata
- semantics

$A=\langle Q, I_0, \mu, E, \tau, \pi\rangle$

valuations

$\nu \models \eta$, ν *satisfies a* **state predicate** η

$\nu \models p$	iff	$\nu(p) = true$
$\nu \models x \sim c$	iff	$\nu(x) \sim c$
$\nu \models \neg\eta_1$	iff	not $\nu \models \eta_1$
$\nu \models \eta_1 \vee \eta_2$	iff	$\nu \models \eta_1$ or $\nu \models \eta_2$

states $(q,\nu) \models \eta$ iff $\nu \models \eta$

37

Timed automata
- semantics

- Given a $\delta \in R^+$, $\nu + \delta$ is a new mapping
  - $(\nu + \delta)(p) = \nu(p)$
  - $(\nu + \delta)(x) = \nu(x) + \delta$
- Given $s = (q,\nu)$, $s + \delta = (q, \nu + \delta)$
- Given a $Y \subseteq X$, νY is a new mapping
  - $(\nu Y)(p) = \nu(p)$
  - $(\nu Y)(x) = 0$ iff $x \in Y$
  - $(\nu Y)(x) = \nu(x)$ iff $x \notin Y$
- Given $s = (q,\nu)$ and $Y \subseteq X$, $sY = (q, \nu Y)$

38

Timed automata
- semantics

$A=\langle Q, I_0, \mu, E, \tau, \pi\rangle$, $s = (q,\nu)$, and $s' = (q',\nu')$

$s \rightarrow s'$ *(there is a (discrete) transitions from s to s')*

iff

- $(q, q') \in E$
- $s \models \tau(q, q')$
- $s\pi(q, q') = s'$
- $s' \models \mu(q')$

39

Timed automata
- semantics

Intuitively, a (linear) computation of a timed automata should be

- **a mapping**
  from R^+ (non-negative reals) to states
- **dense!**

40

Timed automata - semantics

$A=\langle Q, l_0, \mu, E, \tau, \pi\rangle$

s_0-run, a (linear) computation of A

$(s_0,t_0)\ (s_1,t_1)\ (s_2,t_2)\ldots (s_k,t_k)\ldots \ldots \ldots$

- for all $k \geq 0$, $s_k=(q_k, v_k)$
- a nonmonotonically increasing, divergent, non-negative real sequence

  $t_1\ t_2 \ldots t_k \ldots \ldots \ldots$

- for all $k \geq 0$,
  - for all $t\in[0, t_{k+1}-t_k]$, $s_k+t \vDash \mu(q_k)$
  - $s_k+(t_{k+1}-t_k) = s_{k+1}$ or $s_k+(t_{k+1}-t_k) \rightarrow s_{k+1}$

41

Parallel Composition of TA

$A_1=\langle Q_1, I_1, \mu_1, E_1, \tau_1, \pi_1\rangle$,

$A_2=\langle Q_2, I_2, \mu_2, E_2, \tau_2, \pi_2\rangle$

$A_1 \parallel A_2 = \langle Q_1\times Q_2, I_1\wedge I_2, \mu, E, \tau_1\wedge\tau_2, \pi_1\cup\pi_2\rangle$

- $\mu(q, q') = \mu_1(q) \wedge \mu_2(q')$
- E: set of interleaved transitions with synchronization transitions identified

42

Model of Manufacturing System

- *M* = D-Robot || G-Robot || Station || Box
- Transitions with same labels are identified as one in *M*.
- E.g.: g-put in G-Robot, Station, and Box
- E.g.: d-pick in D-Robot, Station, and Box

43

Timed automata - a manufacturing example (1/5)

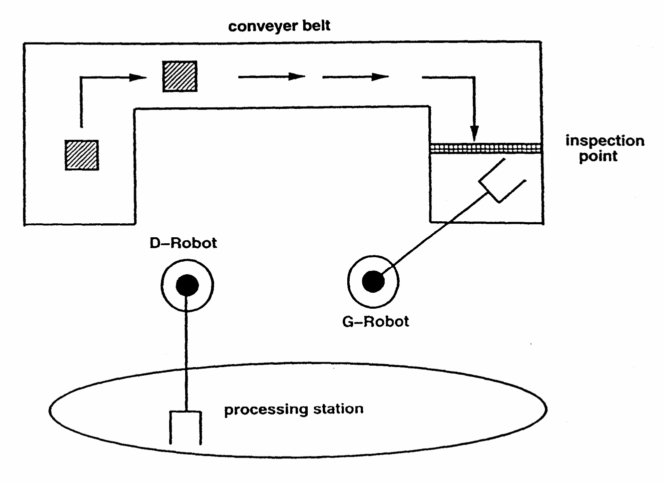


Figure 17.2
A manufacturing example.

44

Timed automata - a manufacturing example (2/5)

D-Robot TA

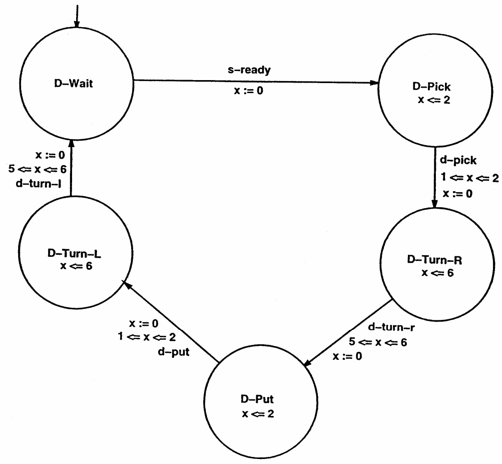


Figure 17.3
Timed automaton for D-Robot.

5

Timed automata - a manufacturing example (3/5)

G-Robot TA

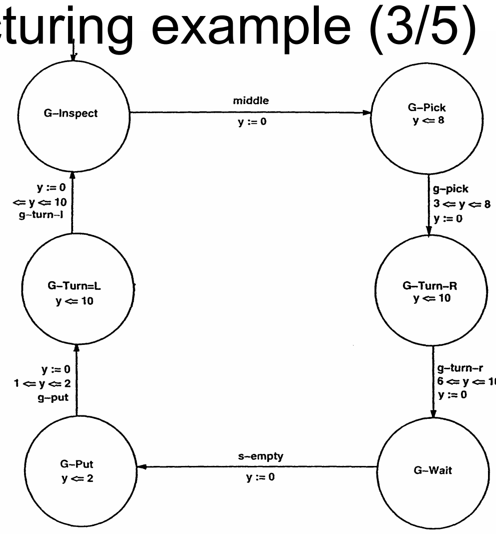


Figure 17.4
Timed automaton for G-Robot.

Timed automata
- a manufacturing example (4/5)

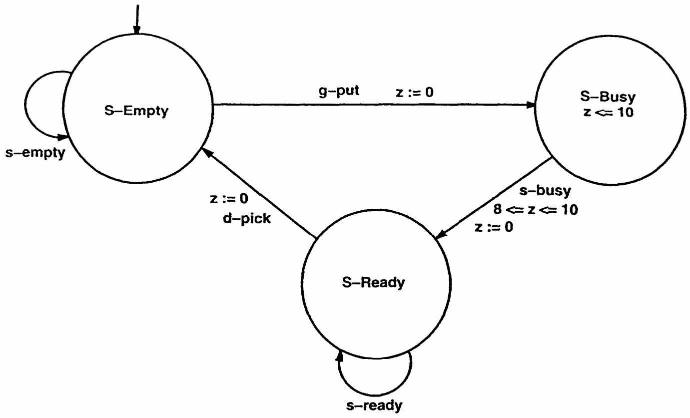


Figure 17.5
Time automaton for processing station.

Station
TA

47

Timed automata
- a manufacturing example (5/5)

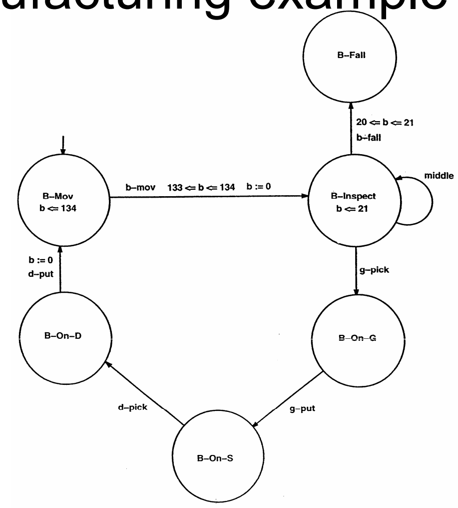


Figure 17.6
Timed automaton for box.

Box TA

48

TCTL (Timed CTL)

<u>Example</u>: ***from now on, it is possible that I will get salary on the 7th day.***

$\exists\Diamond_{=7}$ **salary**

<u>Example</u>: ***from now on, I will be surely married in 10 years.***

$\forall\Diamond_{\leq 10}$ **married**

49

TCTL, continued

<u>Example</u>: ***When a hostile fighter is found, it is always possible that we take it down in 5 sec.***

$\forall\Box$(HF.Found $\rightarrow \exists\Diamond_{\leq 5}$ HF.TakenDown))

<u>Example</u>: ***When a hostile fighter is found, we will always take it down in 5 sec.***

$\forall\Box$(HF.Found $\rightarrow \forall\Diamond_{\leq 5}$ HF.TakenDown))

50

TCTL, continued

Example: **From every state, there is a computation that increment the time divergently.**

$$\forall\Box\exists\Diamond_{=1}\mathit{true}$$

Example: **From every state, time always diverges.**

$$\forall\Box\forall\Diamond_{=1}\mathit{true}$$

51

TCTL, continued

Example: ***After the wedding, I could be happy for 5 days.***

$$\forall\Box(\text{wedding}\rightarrow\exists\Box_{<5}\,\mathit{happy})$$

Example: ***After the wedding, I will be happy for 5 days.***

$$\forall\Box(\text{wedding}\rightarrow\forall\Box_{<5}\,\mathit{happy})$$

52

TCTL
- syntax

BNF

φ ::= η | $\neg\varphi_1$ | $\varphi_1 \vee \varphi_2$ | $\exists \varphi_1 U_{\sim c} \varphi_2$ | $\exists \Box_{\sim c} \varphi_1$

- $\eta \in$ **B(*P*, *X*)**, the set of state predicates of P and X
- $c \in N$;
- '~'$\in \{\leq, \geq, <, >, =\}$

53

TCTL
- syntax

shorthands:

$\varphi_1 \wedge \varphi_2 \equiv \neg((\neg\varphi_1) \vee (\neg\varphi_2))$

$\varphi_1 \rightarrow \varphi_2 \equiv (\neg\varphi_1) \vee \varphi_2$

$\exists \Diamond_{\sim c} \varphi \equiv \exists\, true\, U_{\sim c} \varphi$

$\forall \Box_{\sim c} \varphi \equiv \neg \exists \Diamond_{\sim c} \neg \varphi$

$\forall \varphi_1 U_{\sim c} \varphi_2 \equiv \neg((\exists(\neg\varphi_2) U_{\sim c} \neg(\varphi_1 \vee \varphi_2)) \vee \exists \Box_{\sim c} \neg \varphi_2)$

$\forall \Diamond_{\sim c} \varphi \equiv \forall\, true\, U_{\sim c} \varphi$

54

TCTL
- semantics

$A=\langle Q, I_0, \mu, E, \tau, \pi\rangle$

$A,(q,\nu) \vDash \varphi$, state (q,ν) of A satisfies φ.

$A,(q,\nu) \vDash \eta$ *, as defined for state-predicates*

$A,(q,\nu) \vDash \neg\varphi_1$ iff not $A,(q,\nu) \vDash \varphi_1$

$A,(q,\nu) \vDash \varphi_1 \vee \varphi_2$ iff $A,(q,\nu) \vDash \varphi_1$ or $A,(q,\nu) \vDash \varphi_2$

$A,(q,\nu) \vDash \exists\varphi_1 U_{\sim c}\varphi_2$ iff ...

$A,(q,\nu) \vDash \exists\Box_{\sim c}\varphi_1$ iff ...

55

TCTL
- semantics of $\exists\varphi_1 U_{\sim c}\varphi_2$

$A=\langle Q, I_0, \mu, E, \tau, \pi\rangle$

$A,(q,\nu) \vDash \exists\varphi_1 U_{\sim c}\varphi_2$ iff there are

- a (q,ν)-run:
  $(s_0,t_0)\ (s_1,t_1)\ (s_2,t_2)\ldots(s_k,t_k)\ldots\ \ldots\ \ldots$, and
- a nonmonotonically increasing, divergent, non-negative real sequence: $t_0\, t_1\, t_2 \ldots\ldots$, and
- $k \geq 0$
  - $t_k - t_0 \sim c$
  - $A, s_k \vDash \varphi_2$
  - for all $k > h \geq 0$ and $t' \in [0, t_{h+1} - t_h]$, $A, s_h + t' \vDash \varphi_1$

56

TCTL - semantics of $\exists\Box_{\sim c}\varphi_1$

$A=\langle Q, I_0, \mu, E, \tau, \pi\rangle$

$A,(q,\nu) \models \exists\Box_{\sim c}\varphi_1$ iff there are

- a (q,ν)-run:

  $(s_0,t_0)\ (s_1,t_1)\ (s_2,t_2)\ldots(s_k,t_k)\ldots\ldots\ldots$ and
- a nonmonotonically increasing, divergent, non-negative real sequence: $t_0\, t_1\, t_2\ldots\ldots$

such that for all $k \geq 0$ and $t \in [0, t_{k+1} - t_k]$,

if $t_k + t - t_0 \sim c$, $A, s_k + t \models \varphi_1$

57

TCTL - model-checking problem

A $\models \varphi$, ***A* satisfies** φ, ***A is a model of*** φ

Given a TA $A=\langle Q, I_0, \mu, E, \tau, \pi\rangle$

and a TCTL formula φ,

$A \models \varphi$ **iff** $\forall(q,\nu) \models I_0$, **then** $A,(q,\nu) \models \varphi$

TCTL model-checking problem:

Given a TA $A=\langle Q, I_0, \mu, E, \tau, \pi\rangle$

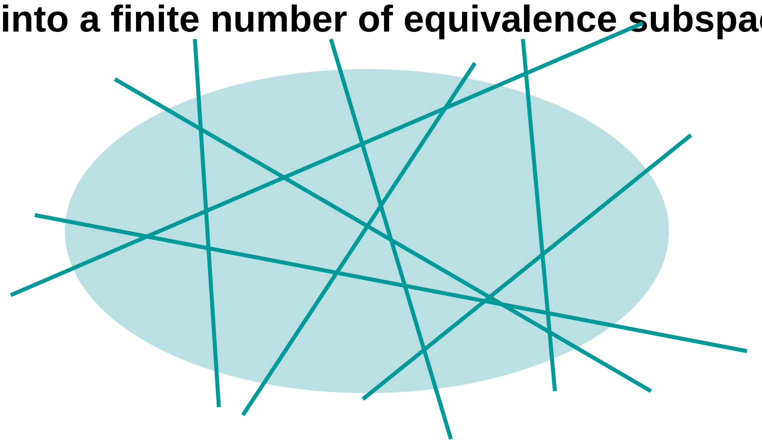
and a TCTL formula φ,

is A a model of φ **?**

58

TCTL
- model-checking problem

How to partition the infinite dense state-space into a finite number of equivalence subspaces ?



59

Region Equivalence [Alur et al 90]

$C_{A:\varphi}$: the biggest timing constant used in *A and* φ.

$(q, \nu) \equiv (q', \nu')$ if and only if

- q = q'
- for all $p \in P$, $\nu(p) = \nu'(p)$
- for all $x \in X$, *when* $\nu(x) \leq C_{A:\varphi}$ or $\nu'(x) \leq C_{A:\varphi}$,
  their integer parts are the same.
  $$\nu(x) \leq C_{A:\varphi} \vee \nu'(x) \leq C_{A:\varphi} \Rightarrow \lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$$
- for all $x,y \in X \cup \{0\}$ ，when $\nu(x)$ or $\nu'(x) \leq C_{A:\varphi}$,
  their fractional parts are of the same ordering.
  $$\nu(x) \leq C_{A:\varphi} \wedge \nu'(x) \leq C_{A:\varphi} \Rightarrow$$
  $$(\nu(x) - \lfloor \nu(x) \rfloor \leq \nu(y) - \lfloor \nu(y) \rfloor \Leftrightarrow \nu'(x) - \lfloor \nu'(x) \rfloor \leq \nu'(y) - \lfloor \nu'(y) \rfloor)$$

60

Region Eq… ur…

In the same control location

The same sampling of Boolean flags

satisfy the same state-predicates

increment their integer parts in the same order.

$C_{A:\varphi}$: the biggest timing constant used in *A and* …
(q, ν)≡(q',ν') if and only if

- q = q'
- for all *p* ∈ *P*, ν(*p*) = ν'(*p*)
- for all *x* ∈ *X*, *when* ν(*x*) ≤ $C_{A:\varphi}$ or ν'(*x*) ≤ $C_{A:\varphi}$, their integer parts are the same.

$$\nu(x) \le C_{A:\varphi} \vee \nu'(x) \le C_{A:\varphi} \Rightarrow \lfloor \nu(x) \rfloor = \lfloor \nu'(\ldots$$

- for all *x,y* ∈ *X* ∪{0}, their fractional parts are of the same ordering.

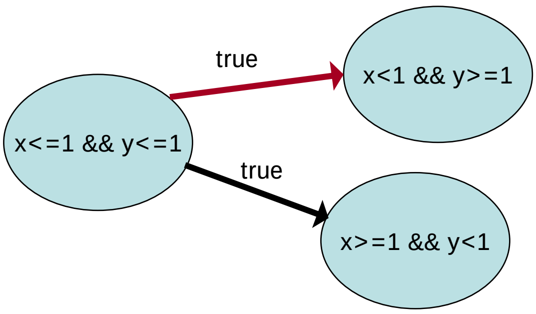
$$\nu(x) \le C_{A:\varphi} \wedge \nu'(x) \le C_{A:\varphi} \Rightarrow$$
$$(\nu(x)-\lfloor \nu(x) \rfloor \le \nu(y)-\lfloor \nu(y) \rfloor \Leftrightarrow \nu'(x)-\lfloor \nu'(x) \rfloor \le \nu \ldots$$

61

Region equivalence Example (1) of invariances

ν(*x*) = 0.5; ν(*y*) = 0.6;

ν'(*x*)=0.599; ν'(*y*)=0.6;

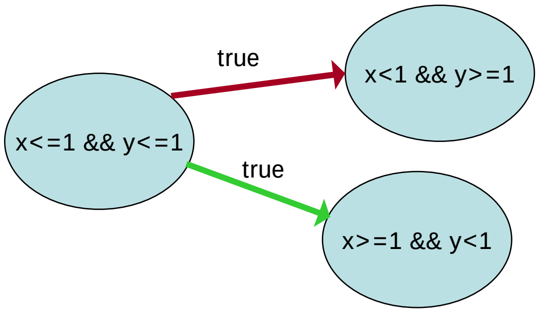


62

Region equivalence
Example (2) of invariances

$\nu(x) = 0.5$; $\nu(y) = 0.6$;

$\nu'(x)=0.8$; $\nu'(y)=0.6$;

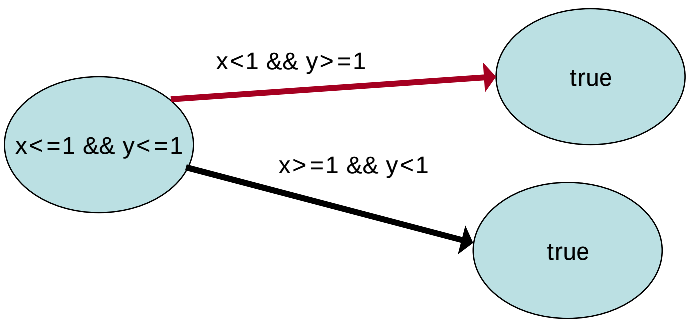


63

Region equivalence
Example (3) of triggers

$\nu(x) = 0.5$; $\nu(y) = 0.6$;

$\nu'(x)=0.599$; $\nu'(y)=0.6$;

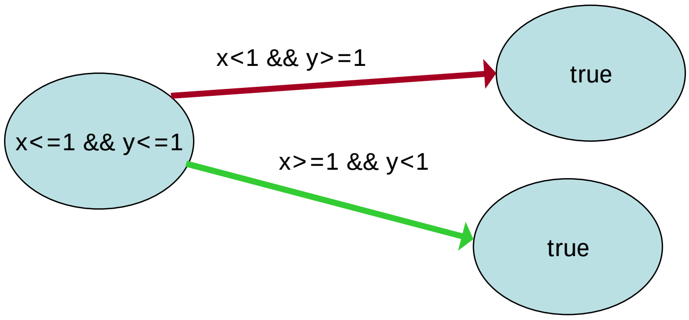


64

Region equivalence
Example (4) of triggers

$\nu(x)$ = 0.5; $\nu(y)$ = 0.6;

$\nu'(x)$=0.8; $\nu'(y)$=0.6;



65

Region Equivalence

$C_{A:\varphi}$**=7,** clocks ***x, y***

(q, ν)	(q, ν')	
$\nu(x)$=3.5 $\nu(y)$=5.5	$\nu'(x)$=3.7 $\nu'(y)$=5.3	**inequivalent**
$\nu(x)$=3.2 $\nu(y)$=5.5	$\nu'(x)$=4.3 $\nu'(y)$=5.8	**inequivalent**
$\nu(x)$=3.1 $\nu(y)$=5.8	$\nu'(x)$=3.5 $\nu'(y)$=5.51	**equivalent**
$\nu(x)$=13.5 $\nu(y)$=8.5	$\nu'(x)$=89 $\nu'(y)$=1003	**equivalent**
$\nu(x)$=5 $\nu(y)$=8.5	$\nu'(x)$=89 $\nu'(y)$=1003	**inequivalent**

66

Region Equivalence Sizes

$C_{A:\varphi}$: **the biggest timing con**[…]**ed in *A and*** φ.

x=5?

choices of flag values

Choices of the integer part of a clock value.

$$2 \cdot |Q| \cdot |2^{P}| \cdot |2^{X}| \cdot (C_{A:\varphi} +1)^{|X|} \cdot |X|^{|X|}$$

choices of control locations

Is a clock ≤ $C_{A:\varphi}$?

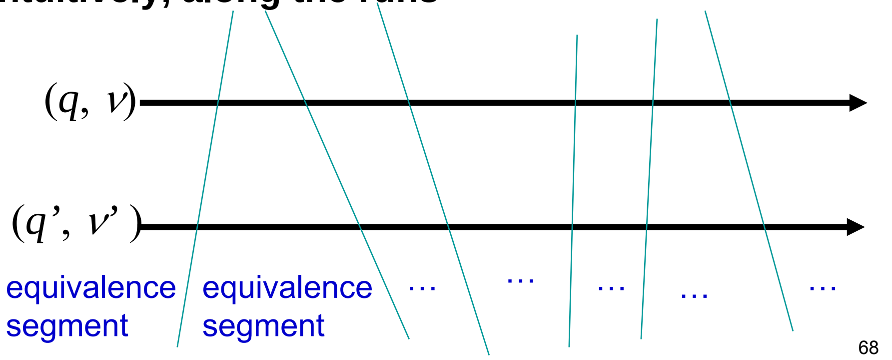
Choices of the ordering of the fractional parts of clock values

TCTL - model-checking problem

Theorem: for all ϕ *with* $C_{A:\phi} \leq C_{A:\varphi}$, *if* $(q, \nu) \equiv (q', \nu')$,

$A, (q, \nu) \models \phi$ iff $A, (q', \nu') \models \phi$

Intuitively, along the runs



68

TCTL - model-checking problem

Region graph (V, F)

- **V: the set of equivalence subspace**
  - partition the state-space into a finite number of equivalence subspaces with the equivalence relatoin $s \equiv s'$.
  - $[s]$: the subspace in which all states $\equiv s$
  - region: a maximal equivalence subspace
- **F: transitions between [*s*], [*s'*]**
  - discrete transitions in the timed automata
  - time-progress of the timed automata

69

TCTL - model-checking problem

to model-check TCTL formulas, we need an auxiliary clock *z* in the region graph

- ***ν*[*z*] is identical to *ν* except *ν*[*z*](*z*)= 0**
- $A,(q,\nu) \vDash \exists \varphi_1 U_{\sim c} \varphi_2$ iff $A,(q,\nu[z]) \vDash \exists \varphi_1 U_{\sim c} \varphi_2$
- $A,(q,\nu[z]) \vDash \exists \varphi_1 U_{\sim c} \varphi_2$ **iff there exists a region sequence $r_0 r_1 r_2 \ldots r_k$ from $(q,\nu[z])$**
  - $r_0 r_1 r_2 \ldots r_{k-1}$ satisfies φ_1
  - r_k satisfies φ_2 and $z \sim c$ 。

70

Region Equivalence How to record a region ?

bits needed for recording a region

$\log (2 \cdot |Q| \cdot |2^P| \cdot |2^X| \cdot (C_{A:\varphi} + 1)^{|X|} \cdot |X|^{|X|})$
$= \log 2 + \log |Q| + |P| \log 2$
$\quad + |X| \log 2 + |X| \log (C_{A:\varphi} + 1) + |X| \log |X|$
$= 1 + \log |Q| + |P| + |X|$
$\quad + |X| \log (C_{A:\varphi} + 1) + |X| \log |X|$

71

TCTL - model-checking problem

- **TCTL model-checking problem is PSPACE-complete. We need**
  - **a counter of log|# region| bits and**
  - **a last-region record of log|# region| bits**
- **TCTL satisfiability problem is undecidable.**

72

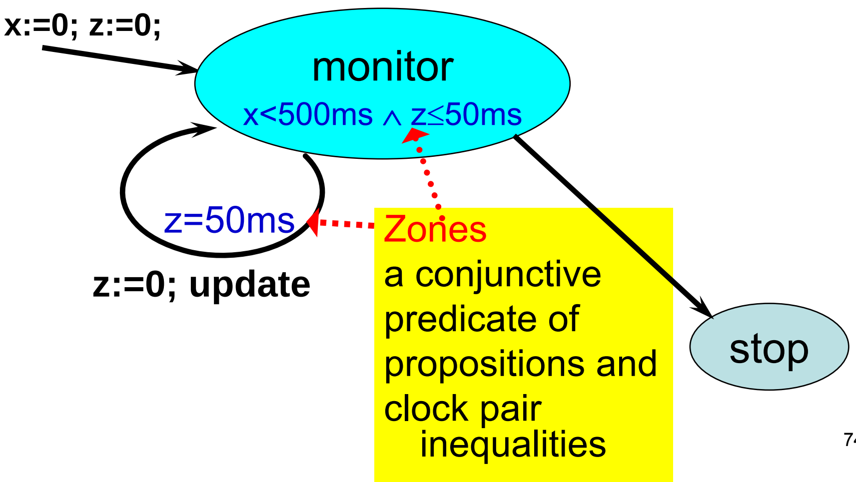
TCTL
- model-checking problem

Workout:

Given a clock set X and a biggest timing constant $C_{A:\phi}$, please derive the complexity upper-bound of region count.

73

Zones in a timed automata



74

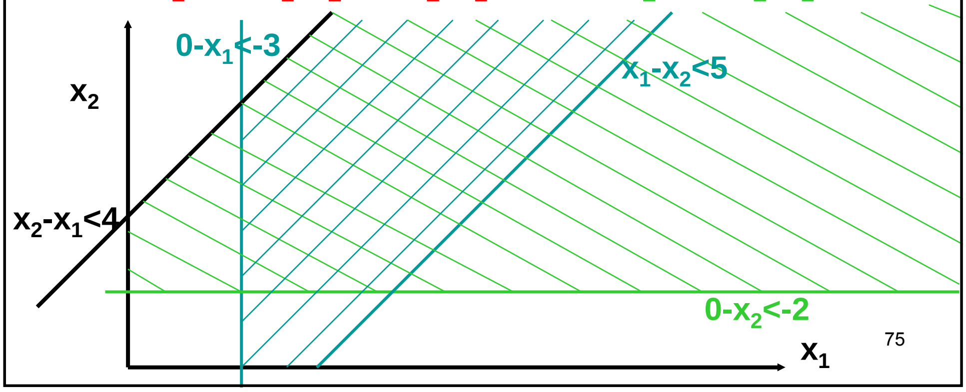
Zones

Two zones

$(0-x_1 \leq -3 \wedge x_1-x_2<5 \wedge x_2-x_1<4) \vee (0-x_2<-2 \wedge x_2-x_1<4)$

Can also be viewed as a union of set of literals,

$\{0-x_1 \leq -3, x_1-x_2<5, x_2-x_1<4\} \cup \{0-x_2<-2, x_2-x_1<4\}$



75

Clock Zones

- Finite representation of infinite state-space
- Conjunction of inequalities such as:

$$x < c \quad | \quad x \leq c \quad | \quad x - y < c \quad | \quad x - y \leq c$$

$$x, y \in X, c \in \mathbf{Z}$$

- General form of a clock zone:

$$x_0 = 0 \wedge \bigwedge_{0 \leq i \neq j \leq n} (x_i - x_j \sim c_{ij})$$

$$\sim \in \{<, \leq\}$$

76

Clock Zones (Operations)

- Clock zones are closed under 3 operations:
- Let z_1, z_2 be two clock zones, $Y \subseteq X$, $t \geq 0$
- **Intersection**: $z_1 \wedge z_2$ is a clock zone
  (Conjunction of conjunctions)
- **Clock Reset**: $z_1(Y{:=}0)$ is a clock zone
- **Time Elapse**: $z_1 + t$ is a clock zone

77

Clock Zones (Time Elapse)

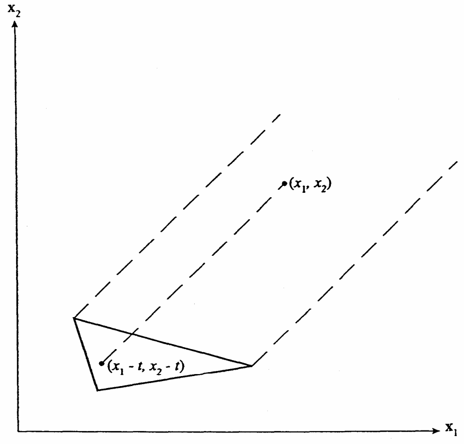


Figure 17.8
The clock zones φ and $\varphi^{\Uparrow}$.

78

Zones = Symbolic States

- **Zone: (*s*, *z*) , where *s*: location, *z*: clock zone represents a symbolic state.**
- **Predecessor Clock Zone: *z*' = pred(*z*, *e*), where *z*: clock zone, *e*: transition (predecessor clock zone obtained from *z* before time elapse and executing transition *e*)**
- **Predecessor Zone: (*s*', pred(*z*, *e*)), where *e*: *s*→*s*'**

79

Clock Zone (succ(*z*, *e*))

To obtain Predecessor Clock Zone (pred(*z*, *e*))

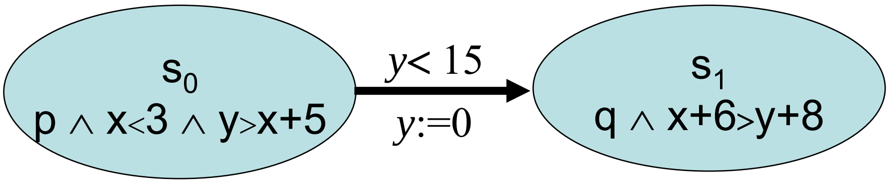
- **Reset all clocks from $\pi(e)$**
- **Intersect with $\tau(e)$**
- **Intersect with $\mu(s)$**
- **Let time elapse in *s* (operator te())**
- **Intersect *z* with $\mu(s)$**

pred(*z*, *e*) = tbck((*z* [π :=0]) $\wedge$ τ(*e*) $\wedge \mu$ (s)) $\wedge$ μ(s)

Closed under $\wedge$, tbck(), reset, also a clock zone!!!

80

TCTL model-checking
- symbolic algorithms



In s_0,

- **what is the weakest precondition for the transition ?**

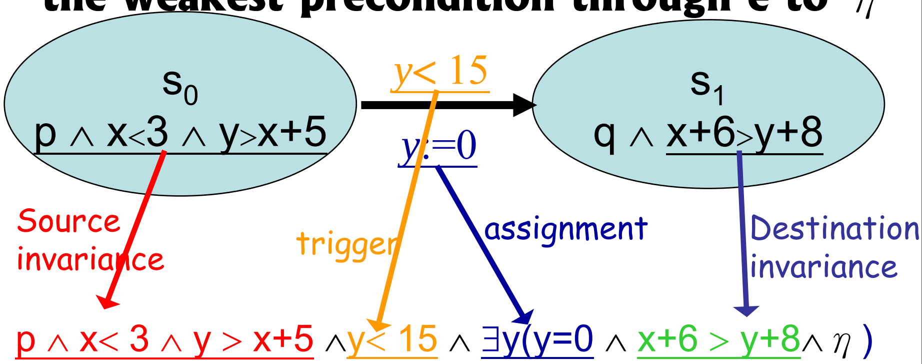
  xtion_bck$_e$(η): the weakest precondition through e to η ?

- **what is the weakest precondition to stay in s_0 through time progress $\delta \geq 0$?**

  time_bck(η): the weakest precondition through time progress to η ?

81

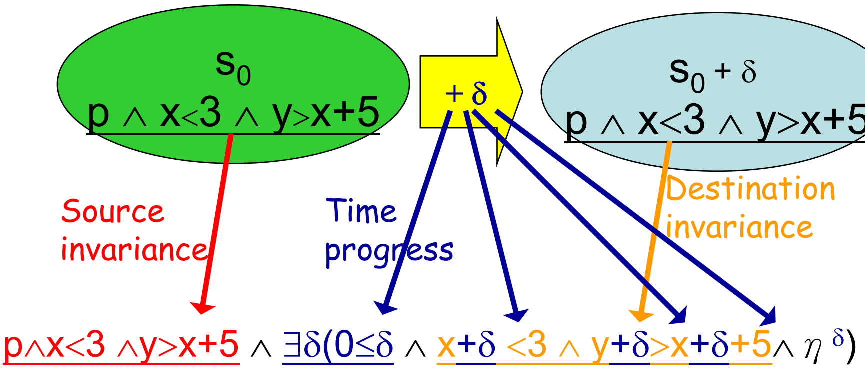
xtion_bck$_e$(η)
- the weakest precondition through e to η



How to get rid of the $\exists y$?

82

time_bck(η)
- the weakest precondition through time progress to η



How to get rid of the ∃δ ?

83

How to get rid of the ∃δ ?

$$\exists\delta\ (2\text{-}x < \delta \wedge \delta < 15\text{-}y) \equiv 2\text{-}x<15\text{-}y$$

- Density of reals!
- Projections in geometry.

Joseph Fourier, 1824!
(1768-1830)

84

How to get rid of the $\exists\delta$?

Assume that $\eta = 2\text{-}x < 0 \wedge 0 < 15\text{-}y$

s_0: $p \wedge x<3 \wedge y>x+5$ → $+\delta$ → s_0: $p \wedge x<3 \wedge y>x+5$

pairwise matching, eliminate δ with transitivity

$p \wedge y > x + 5 \wedge \exists\delta\,(0 \leq \delta \wedge 2\text{-}x < \delta \wedge \delta < 15\text{-}y \wedge \delta < 3\text{-}x)$

independent of δ — ***δ to the right*** — ***δ to the left***

$0 \leq \delta$ $\quad$ $\delta < 15\text{-}y$ $\quad$ ⇒ $\quad$ $0 < 15\text{-}y$, $0 < 3\text{-}x$

$2\text{-}x < \delta$ $\quad$ $\delta < 3\text{-}x$ $\quad$ $2\text{-}x < 15\text{-}y$, $2\text{-}x < 3\text{-}x$

thus we get $p \wedge y > x + 5 \wedge y<15 \wedge x< 3 \wedge y\text{-}x<13$

85

TCTL model-checking - symbolic algorithms

The weakest precondition of time-progress in s_0: $x\text{-}2 > 0 \wedge y< 15 \wedge p \wedge x< 3 \wedge y > x+5$?

$s_0 + \delta$: $p \wedge x<3 \wedge y>x+5$ → $+\delta$ → s_0: $x\text{-}2>0 \wedge y<15 \wedge p \wedge x<3 \wedge y>x+5$

With the invariance condition of s_0,

$p \wedge y>x + 5 \wedge y<15 \wedge x<3 \wedge y\text{-}x<13 \wedge$ $\boldsymbol{p \wedge x<3 \wedge y>x+5}$

we still get

$p \wedge y > x + 5 \wedge y<15 \wedge x< 3 \wedge y\text{-}x<13$

86

TCTL model-checking - symbolic algorithms

Safety analysis: Given initial condition I and risk condition $\neg\eta$,

How to evaluate if **$\neg\eta$ can happen ?**

S:= $\neg\eta$ ***; S':=false;***

while S $\neq$ ***S' {***

S':=S;

S:= S $\vee$ *time_bck* ($\vee_{e \in T}$ *xtion_bck* (*S, e*)) ;

}

return I $\wedge$ ***S*** **;**

87

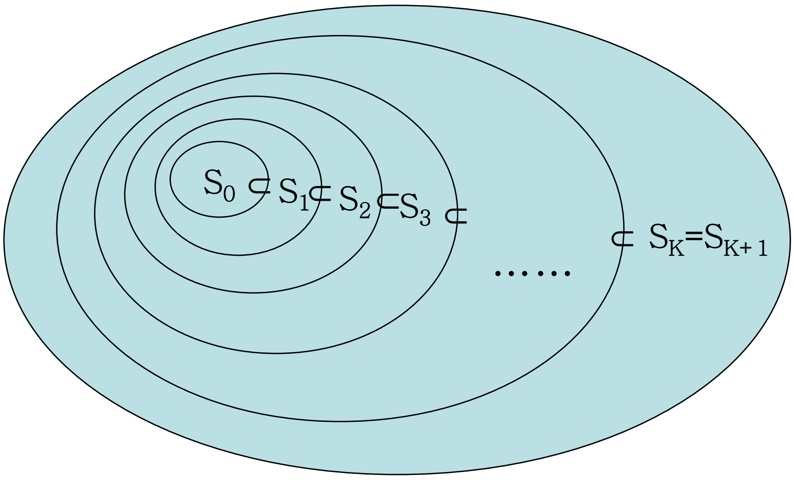
Zone Graph

- **Zone Graph is a transition system *Z*(*A*)**
- **States = zones of *A***
- **Initial state = (s, [X := 0])**
- **For each transition *e* of A:**

  a transition: (*s*, *z*) → (*s*', succ(*z*, *e*))
- **Zone reachability ➔State reachability**

88

Backward Reachability Analysis
- *Least fixpoint calculation*



89

Compute $\exists x(\ f_1(x_1, ..., x_n)<x \wedge ...\wedge f_h(x_1, ..., x_n)<x$
$\wedge \ x< g_1(x_1, ..., x_n)\wedge...\wedge x< g_k(x_1, ..., x_n))$

x in the upper-bound

An important technique:

with transitivity, pairwise match the inequalities with x

$f_1(x_1, ..., x_n)< x \wedge f_2(x_1, ..., x_n)< x \wedge ...\wedge f_h(x_1, ..., x_n)< x$
$\wedge \ x< g_1(x_1, ..., x_n)\wedge x< g_2(x_1, ..., x_n)\wedge...\wedge x< g_k(x_1, ..., x_n)$

x in the lower-bound

The condition for a solution to x

$\forall \ 1\leq i \leq k \ \forall \ 1\leq j \leq k \ (f_i(x_1, ..., x_n)< g_j(x_1, ..., x_n))$

i、j pairwise match

90

Compute $\exists x(\ f_1(x_1, \dots, x_n)<x \wedge \dots \wedge f_h(x_1, \dots, x_n)<x \wedge x< g_1(x_1, \dots, x_n) \wedge \dots \wedge x< g_k(x_1, \dots, x_n))$

An important technique:

with transitivity, pairwise match the inequalities with x

$f_1(x_1, \dots, x_n)< x \wedge f_2(x_1, \dots, x_n)< x \wedge \dots \wedge f_h(x_1, \dots, x_n)< x$
$\wedge\ x< g_1(x_1, \dots, x_n) \wedge x< g_2(x_1, \dots, x_n) \wedge \dots \wedge x< g_k(x_1, \dots, x_n)$

In geometry, this is the projection of a convex hull in n+1 dimension space of $x, x_1, \dots, x_n$ **on the n dimension space of** $x_1, \dots, x_n$

91

Compute $\exists x(\ f_1(x_1, \dots, x_n)<x \wedge \dots \wedge f_h(x_1, \dots, x_n)<x \wedge x< g_1(x_1, \dots, x_n) \wedge \dots \wedge x< g_k(x_1, \dots, x_n))$

In symbolic computation, an equivalence space is described with combinational inequalities.

Example: $x< 3 \wedge y > x+5 \wedge z \le x+5$

This is a convex hull surounded by the planes of $x=3$, $y = x+5$, **and** $z = x+5$ **in the 3-dimension space.**

Example: $x< 3 \wedge y > x+5 \wedge z \le w +5$

This is a convex hull surounded by the three planes of $x=3$, $y = x+5$, $z = w+5$ **in the 4-dimension space.**

92

Zones

$(0-x_1 \le -3 \wedge x_1-x_2<5 \wedge x_2-x_1<4) \vee (0-x_2<-2 \wedge x_2-x_1<4)$

Normal forms

- *closure form:* ***all-pair shortest-path form***

  $(0-x_1 \le -3 \wedge 0-x_2<2 \wedge x_1-x_2<5 \wedge x_2-x_1<4)$

  $\vee (0-x_2<-2 \wedge 0-x_1<2 \wedge x_2-x_1<4)$

  - ***always the most number of constraints***

- *reduced form:* ***minimum number of constraints***

  $(0-x_1 \le -3 \wedge x_1-x_2<5 \wedge x_2-x_1<4)$

  $\vee (0-x_2<-2 \wedge x_2-x_1<4)$

93

Data-structures for Zones - Difference Bound Matrix (DBM)

- DBM is a matrix to represent a clock zone

	0	x_1	$\dots$	x_n	
0		$< c$			$0 - x_1 < c$, i.e., $x_1 > c$
x_1	$< \infty$			$\le r$	$x_1 < \infty$ $x_1 - x_n \le r$
$\dots$					
x_n	$\le d$				$x_n \le d$

94

Data-structures for Zones - DBM (Example)

- $x_1 - x_2 < 2 \quad \wedge \quad 0 < x_2 \leq 2 \quad \wedge \quad 1 \leq x_1$

	0	x_1	x_2
0	≤ 0	≤ -1	< 0
x_1	$< \infty$	≤ 0	< 2
x_2	≤ 2	$< \infty$	≤ 0

95

Data-structures for Zones - DBM (Uniqueness)

- A zone is not uniquely represented by a DBM
- Zone: $x_1 - x_2 < 2 \quad \wedge \quad 0 < x_2 \leq 2 \quad \wedge \quad 1 \leq x_1$
- $x_1 - x_2 < 2$ and $x_2 \leq 2 \quad \Rightarrow \quad x_1 < 4$

	0	x_1	x_2
0	≤ 0	≤ -1	< 0
x_1	$< \infty$	≤ 0	< 2
x_2	≤ 2	$< \infty$	≤ 0

⇨

	0	x_1	x_2
0	≤ 0	≤ -1	< 0
x_1	< 4	≤ 0	< 2
x_2	≤ 2	$< \infty$	≤ 0

96

Data-structures for Zones - DBM (Canonical Form)

- **Canonical (unique) form of DBM for a zone**
- **Tightening operation:**

$x_i - x_j \sim_{ij} d_{ij}$ **and** $x_j - x_k \sim_{jk} d_{jk}$ ➔ $x_i - x_k \sim_{ik} d_{ik}$

where $d_{ik} = d_{ij} + d_{jk}$

$\sim_{ik} = \leq$ **if both** $\sim_{ij}$ **and** $\sim_{jk}$ **are** $\leq$

$<$ **otherwise**

- **Apply tightening operations to a DBM until no more change is possible!**

97

Data-structures for Zones - DBM (Emptiness)

- **Check if all elements on main diagonal are (≤ 0)**
  - Yes ➔ nonempty
  - No ➔ empty or unsatisfiable
- **Empty or unsatisfiable ➔ At least 1 negative entry on main diagonal**
- **E.g.** $x_i - x_i \leq -1$ ➔ $0 \leq -1$ ➔ **FALSE!!!**

98

Data-structures for Zones
- DBM (3 operations: Intersection)

- **Intersection:** $D = D_1 \wedge D_2$ **(all DBMs)**
- **Let** $D_1(i, j) = \sim_1 c_1$ **and** $D_2(i, j) = \sim_2 c_2$

$$D(i, j) = (\min(c_1, c_2), \sim)$$

where $\sim = \sim_1$ **if** $c_1 < c_2$

$\sim = \sim_2$ **if** $c_2 < c_1$

$\sim = \sim_1$ **if** $c_1 = c_2$ **and** $\sim_1 = \sim_2$

$\sim = <$ **if** $c_1 = c_2$ **and** $\sim_1 \neq \sim_2$

99

Data-structures for Zones
- DBM (3 operations: Clock Reset)

- **Clock Reset:** $D' = D\,[Y := 0],\ \ Y \subseteq X$
  defined as follows:
  - D'(i, j) = ($\leq$ 0) if $x_i, x_j \in Y$
  - D'(i, j) = D(0, j) if $x_i \in Y$ and $x_j \notin$ Y
  - D'(i, j) = D(i, 0) if $x_j \in Y$ and $x_i \notin$ Y
  - D'(i, j) = D(i, j) if $x_j \notin Y$ and $x_i \notin$ Y

100

Data-structures for Zones - DBM (3 operations: Time Elapse)

- **Time Elapse**: $D' = \mathrm{te}(D)$ defined as follows:
- $D'(i, 0) = (< \infty)$, for any $i \neq 0$
- $D'(i, j) = D(i, j)$, for $i = 0$ or $j \neq 0$

	0	x_1	x_2
0	≤ 0	≤ -1	< 0
x_1	< 4	≤ 0	< 2
x_2	≤ 2	$< \infty$	≤ 0

⇒

	0	x_1	x_2
0	≤ 0	≤ -1	< 0
x_1	$< \infty$	≤ 0	< 2
x_2	$< \infty$	$< \infty$	≤ 0

101

Data-structures for Zones - DBM (3 operations)

- All 3 operations can be **efficiently** implemented
- DBM must be **canonalized** (using tightening) before any of the 3 operations (intersection, clock reset, and time elapse)
- After any of the 3 operations, a DBM might **no longer** be **canonical**!
- **Last step: Reduce to canonical form!!!**

102

Data-structures for Zones - DBM (Zone Graph Construction)

- **Clock zones: represented by DBM**
- **Succesor clock zones succ(*z*, *e*): computed by the 3 operations: intersection, reset, and time elapse on DBM in**

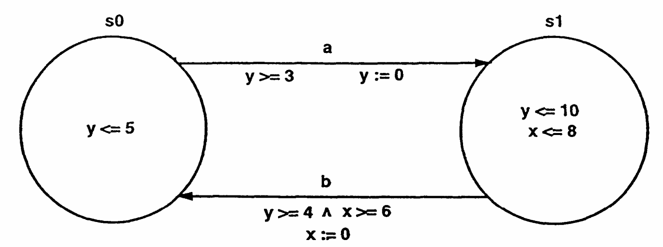


Figure 17.1
A simple timed automaton.

Data-structures for Zones - DBM (Zone Graph Construction)

- **Initial state: (s0, Z0), Z0: $x = 0 \wedge y = 0$**

	0	x	y
0	≤ 0	≤ 0	≤ 0
x	≤ 0	≤ 0	≤ 0
y	≤ 0	≤ 0	≤ 0

Zone D_0

104

Data-structures for Zones - DBM (Zone Graph Construction)

- **Invariant μ (s0) is $0 \leq x \wedge 0 \leq y \leq 5$**

	0	x	y
0	≤ 0	≤ 0	≤ 0
x	$< \infty$	≤ 0	$< \infty$
y	≤ 5	≤ 5	≤ 0

105

Data-structures for Zones - DBM (Zone Graph Construction)

- **Step 1: Intersection D_0 with μ (s0) gives D_0**
- **Step 2: Let time elapse: te($D_0 \wedge \mu$ (s0))**

	0	x	y
0	≤ 0	≤ 0	≤ 0
x	≤ 0	≤ 0	≤ 0
y	≤ 0	≤ 0	≤ 0

⇨

	0	x	y
0	≤ 0	≤ 0	≤ 0
x	$< \infty$	≤ 0	≤ 0
y	$< \infty$	≤ 0	≤ 0

106

Data-structures for Zones
- DBM (Zone Graph Construction)

- **Step 3: Intersect with μ (s0) again**

	0	x	y
0	≤ 0	≤ 0	≤ 0
x	$< \infty$	≤ 0	≤ 0
y	$< \infty$	≤ 0	≤ 0

⇨

	0	x	y
0	≤ 0	≤ 0	≤ 0
x	≤ 5	≤ 0	≤ 0
y	≤ 5	≤ 0	≤ 0

107

Data-structures for Zones
- DBM (Zone Graph Construction)

- **Trigger τ(a) = $y \geq 3$**

	0	x	y
0	≤ 0	≤ 0	≤ -3
x	$< \infty$	≤ 0	$< \infty$
y	$< \infty$	$< \infty$	≤ 0

108

Data-structures for Zones - DBM (Zone Graph Construction)

- **Step 4: Intersect with trigger τ(a)**

	0	x	y
0	≤ 0	≤ 0	≤ 0
x	≤ 5	≤ 0	≤ 0
y	≤ 5	≤ 0	≤ 0

⇒

	0	x	y
0	≤ 0	≤ -3	≤ -3
x	≤ 5	≤ 0	≤ 0
y	≤ 5	≤ 0	≤ 0

109

Data-structures for Zones - DBM (Zone Graph Construction)

- **Step 5: Reset clock *y* in DBM**

	0	x	y
0	≤ 0	≤ -3	≤ -3
x	≤ 5	≤ 0	≤ 0
y	≤ 5	≤ 0	≤ 0

⇒

	0	x	y
0	≤ 0	≤ -3	≤ -3
x	≤ 5	≤ 0	≤ 5
y	≤ 0	≤ -3	≤ 0

$Z1 = 3 \le x \le 5 \wedge 3 \le x - y \le 5 \wedge y = 0$

110

Data-structures for Zones - DBM (Zone Graph Construction)

- **Successor of (s0, Z0) is (s1, Z1)**
- **Repeat the same 5 steps:**
- **(s0, $4 \le y \le 5 \wedge 4 \le y - x \le 5 \wedge x = 0$)**
- **(s1, $0 \le x \le 1 \wedge 0 \le x - y \le 1 \wedge y = 0$)**
- **(s0, $5 \le y \le 8 \wedge 5 \le y - x \le 8 \wedge x = 0$)**
- **(s1, $x = 0 \wedge y = 0$),**
  which is contained in the 2nd zone, thus no more new zones can be generated!!!
  No more change! Stop! Zone Graph!

111

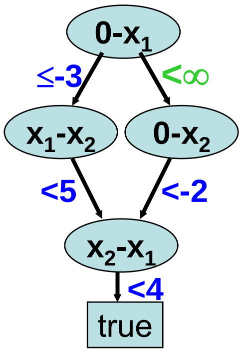
Data-structures for Zones - CRD: Clock-Restriction Diagram

- **A BDD-like data-structure**
- **Recording device for (zone) DBM set**
- **variables like x-x'**
- **Arc values like (<, d), d $\in[-C_A, C_A]\cup\{\infty\}$; or**
  (≤, d), d $\in[-C_A, C_A]$
- **Default value on arcs: (<, ∞)**
  – No constraint!

112

Data-structures for Zones - CRD Example

$(0-x_1 \le -3 \wedge x_1-x_2<5 \wedge x_2-x_1<4) \vee (0-x_2<-2 \wedge x_2-x_1<4)$



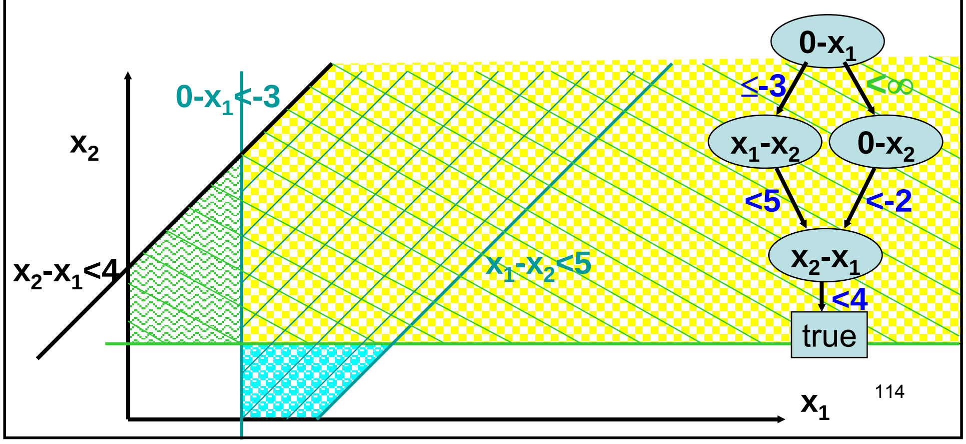
- Each path represents a zone, which is convex.
- The whole CRD represents a concave state-space.
- No canonical anymore.
- <∞ can be used sometimes.

113

Data-structures for Zones - *Representation fragmentation of CDD*

Two zones

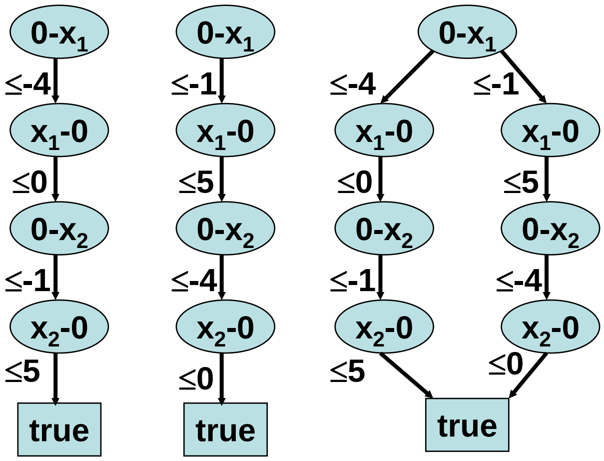
$(0-x_1 \le -3 \wedge x_1-x_2<5 \wedge x_2-x_1<4) \vee (0-x_2<-2 \wedge x_2-x_1<4)$



114

Data-structures for Zones
- *CRD union is simple.*

Example: $(0 \le x_1 \le 4 \wedge 1 \le x_2 \le 5) \vee (1 \le x_1 \le 5 \wedge 0 \le x_2 \le 4)$



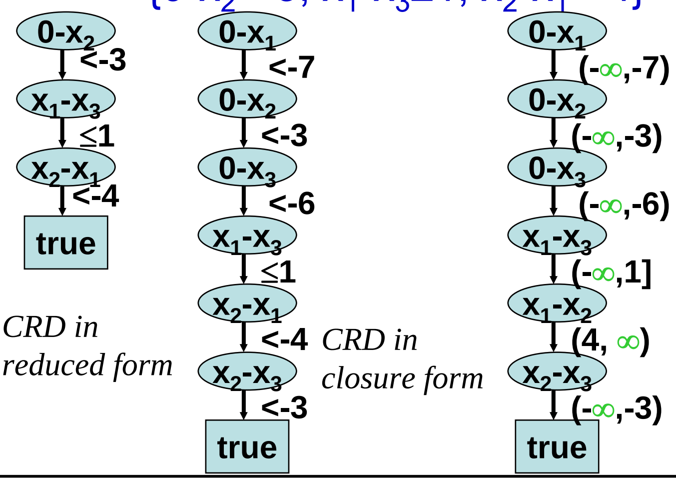
- ***Linear to the sizes of components***
- ***Preserve zone structures***

115

Data-structures for Zones
- *CRD is not canonical!*

In general, reduced CRDs have much fewer nodes along each path!

$\{0-x_2<-3,\ x_1-x_3 \le 1,\ x_2-x_1<-4\}$



CRD in reduced form

CRD in closure form

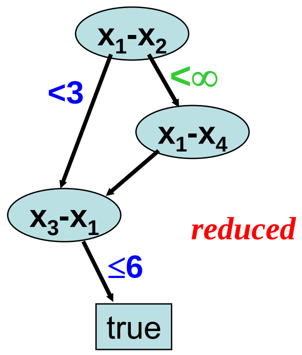
CDD in closure form

116

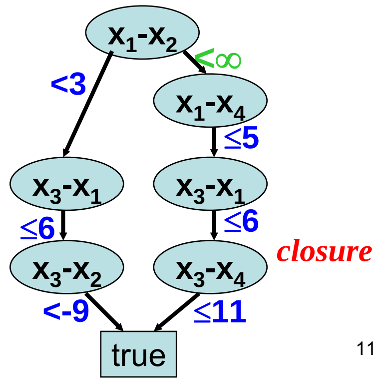
Data-structures for Zones - *Sensitivity to choice of normal forms*

Reduced CRDs are less likely to interfere data-sharing!

(x1-x2<3∧ x3-x1≤6)
∨ (x1-x4≤5∧ x3-x1≤6)



(x1-x2<3 ∧ x3-x1≤6 ∧ x3-x2≤6)
∨(x1-x4≤5 ∧ x3-x1≤6 ∧ x3-x4≤11)



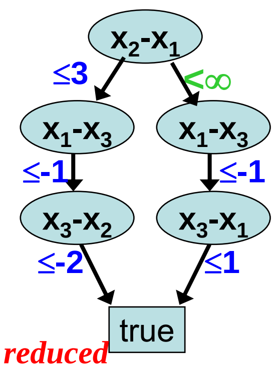
117

Data-structures for Zones - *Sensitivity to choice of normal forms*

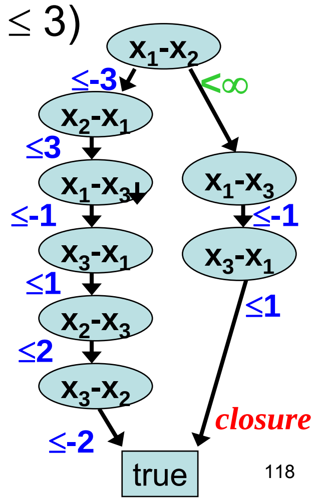
Reduced form makes it difficult to detect zone-containment.

$(x_1-x_3 \leq -1 \wedge x_3-x_2 \leq -2 \wedge x_2-x_1 \leq 3)$

$\vee\ (x_1-x_3 \leq -1 \wedge x_3-x_1 \leq 1)$



Note, it is $O(n^3)$ to deduce the all-pair shortest-path relation.

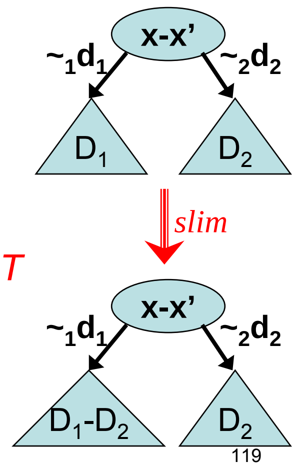


118

Data-structures for Zones
- *Contained zone-path elimination*

- A node with single outgoing arc labeled <∞ can be bypassed.
- Given two arcs, when
  - $\sim_1 d_1$ more restrictive than $\sim_2 d_2$
  - $D_1 \subseteq D_2$

  then D1 can be removed.



★ *The operation MAY or MAY NOT lead to smaller CRD sizes.*

★ *Don't know how to do this with CDD.*

119

Data-structures for Zones
- *Set-oriented manipulations on CRDs*

Given two CRDs D_1:{ζ_1, ζ_2} and D_2:{ζ_2, ζ_3},

- $D_1 \cap D_2$ is the CRD for {ζ_2} $O(|D_1|\times|D_2|)$
- $D_1 \cup D_2$ is the CRD for {$\zeta_1, \zeta_2, \zeta_3$} $O(|D_1|\times|D_2|)$
- D_1 - D_2 is the CRD for {ζ_1} $O(|D_1|\times|D_2|)$

Space-intersection: $D_1 \wedge D_2$

- For every $\zeta_1(x,x') = (\sim_1, d_1)$ and $\zeta_2(x,x') = (\sim_2, d_2)$

$$\zeta_1 \wedge \zeta_2(x,x') = \begin{cases} (\sim_1, d_1) & \text{if } d_1 < d_2 \vee (d_1 == d_2 \wedge \sim == \text{"<"}) \\ (\sim_2, d_2) & \text{otherwise} \end{cases}$$

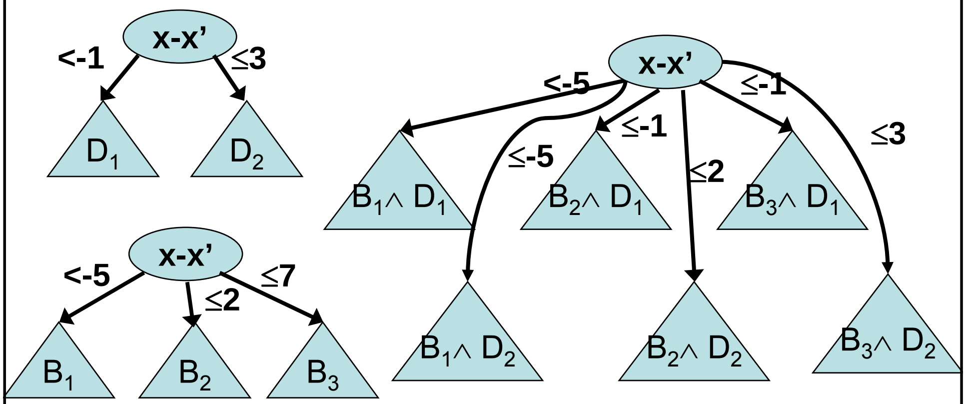
- $D_1 \wedge D_2 = \{\zeta_1 \wedge \zeta_2 \mid \zeta_1 \in D_1; \zeta_2 \in D_2\}$ $O(|D_1|^2\times|D_2|^2)$

120

Data-structures for Zones

- *How to translate the mathematics to procedures on CRDs ?*

Space-intersection: $D \wedge B = \{\zeta_d \wedge \zeta_b \mid \zeta_d \in D; \zeta_b \in B\}$



121

Data-structures for Zones

- Style of CRD manipulating algorithm

set Ψ; /* database of already-processed cases */

$\cup$(B,D) { $\Psi = \emptyset$; return rec$\cup$(B,D); }

rec$\cup$(B, D) with B=(x_B-x_B',(β_i, B_i)$_{1\leq i\leq n}$),D =(x_D-x_D',(α_j, D_j)$_{1\leq j\leq m}$) {

  if B==true, return true; else if D is true, return true;

  else if $\exists$H,(B,D,H)$\in\Psi$, return H; /* efficiency on common structures */

  else if x_B-x_B' precedes x_D-x_D', H:=(x_B-x_B' , (β_i, rec$\cup$(B_i,D))$_{1\leq i\leq n}$);

  else if x_D-x_D' precedes x_B-x_B', H:=(x_D-x_D' , (α_j, rec$\cup$(B, D_j))$_{1\leq j\leq m}$);

  else {

    for (i=n, j= m, H=false; i$\geq$1$\wedge$j $\geq$ 1, do {

      if β_i == α_j , { H= H $\cup$ (x_B-x_B', (β_i , rec$\cup$(B_i,D_j))); i--; j--; }

      else if β_i< α_j, { H= H $\cup$ (x_B-x_B', (α_j, D_j)); j--; }

      else if β_i> α_j, { H= H $\cup$ (x_B-x_B', (β_i , B_i)); i--; }

    }

    if I $\geq$ 1, H= H $\cup$ (x_B-x_B', (β_h , B_h)$_{1\leq h\leq i}$);

    if j $\geq$ 1, H= H $\cup$ (x_B-x_B', (α_h , D_h)$_{1\leq h\leq j}$);

  }

  $\Psi = \Psi \cup$ {(B,D,H)}; return H; /* saving results for common structures */

}

122

Data-structures for Zones - *BDD+CRD*

Can combine BDD with CRD in the same data-structure.

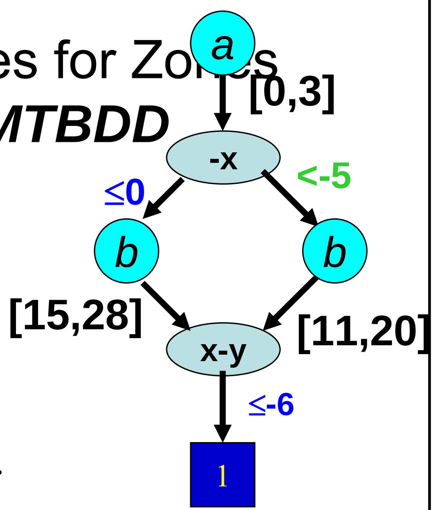
- $D_1 \cap D_2$ is like $D_1 \wedge D_2$
- $D_1 \cup D_2$ is like $D_1 \vee D_2$
- $D_1 - D_2$ is like $D_1 \wedge \neg D_2$

with integrated evaluation ordering.

123

Data-structures for Zones - *CRD+MTBDD*

- Case analysis in the procedures for MTBDD and CRD
- One integrated variable ordering for MTBDD variables and CRD variables.



discrete a, b: 0..30;

clock x, y;

$0 \le a \le 3 \wedge (-x \le 0 \wedge 15 \le b \le 28 \vee -x < -5 \wedge 11 \le b \le 20) \wedge x-y \le -6$

124

Data-structures for Zones

- *Exercise: Costruction of a CRD+MTBDD*

discrete a, b:0..100.

clock x_1, x_2, x_3;

$(a = 3 \wedge x_1 - x_3 \leq -1 \wedge x_3 - x_2 \leq -2$

$\wedge\, b \in [10,20] \wedge x_2 - x_1 \leq 3)$

$\vee\, (a \in [1,5] \wedge b \in [16,20] \wedge x_1 - x_3 \leq -1 \wedge x_3 - x_1 \leq 1)$

Assume the variable ordering:

$a \angle x_1 - x_3 \angle x_3 - x_2 \angle b \angle x_2 - x_1 \angle x_3 - x_1$

125

Workout *for symbolic verification of timed automata*

Given transition (q,q'): when *x* $\leq$ *11* may *y:= 0*;

μ(q): *4 $\leq$ y $\wedge$ y < 25,*

and *ϕ = 9 < x $\wedge$ x-y $\leq$ 7 $\wedge$ y $\leq$ 5*

please calculate

- ***xtion_bck* (ϕ, (q, q'))**
- ***time_bck* (*xtion_bck* (ϕ, (q,q')))**

126

TCTL inevitability analysis

- *Safety properties:* ∀□φ
  - Negation: reachability properties: ∃◊φ
    - **Least fixpoint evaluation**
  - Heavily researched for efficient evaluation
  - nonZeno requirement, not very necessary
- *Inevitability properties:* ∀◊φ
  - Kind of parallel to *liveness* properties in LTL
  - Negation: ∃□φ
    - **Greatest fixpoint evaluation**
  - Not very much researched for efficient evaluation
  - nonZeno requirement necessary

127

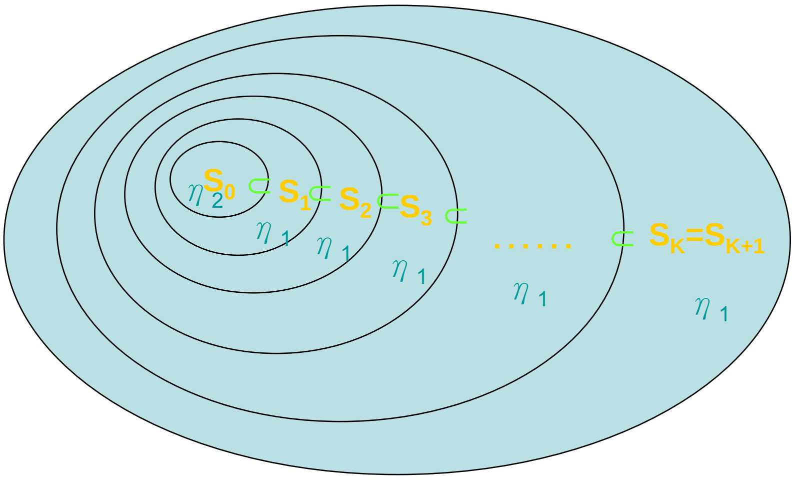
Model checking with Non-Zeno requirement (1)

- Basic procedures
  - Xtion_bck(η ,e)
    - weakest precondition of discrete transitions
  - Time_bck(η)
    - backward time-progression
- Reachable-bck(η_1, η_2) $\equiv$ lfpY.($\eta_2 \vee (\eta_1 \wedge$time_bck($\eta_1 \wedge \bigvee_{e \in T}$xtion_bck(Y,e))))

The lfp (least fixpoint) of F(Y) is the minimal solution to Y=F(Y).

128

Reachable-bck(η_1, η_2)
- *Least fixpoint calculation*



129

Model checking with Non-Zeno requirement (2)

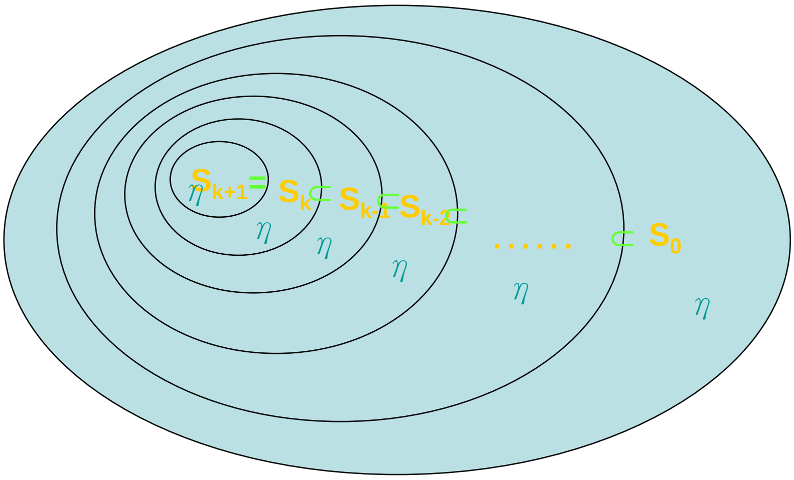
- Lemma: given d≥1, A,

  ν ⊨∃□ η iff there is a finite run ρ
  - from ν
  - of duration ≥d ≥1
  - along ρ every state satisfies η and
  - ρ ends at a state satisfying ∃□ η
- ∃□ η ≡gfp Y.(ZC.reachable-bck(η ,Y∧ZC≥d))

The gfp (greatest fixpoint) of F(Y) is the maximal solution to Y=F(Y).

130

∃□ η
- *Greatest fixpoint calculation*



131

Gfp procedure

∃□ η ≡gfp Y.(ZC.reachable-bck(η ,Y∧ZC≥d))

```
gfp( η ){
    Y:= η ; Y':=true;
    Repeat until Y=Y',{
      Y':=Y;
      Y:=Y∧∃ZC  ( ZC=0∧reachable-bck( η ,Y∧ZC≥d));
    }
    return Y;
 }
```

132

EDGF
- early decision on GFP evaluation

- Observation
  - The state space shrinks iteratively
- Basic idea
  - Stop at a gfp iteration if already no target states are in the gfp.
- Cost
  - Small extra computation

133

EDGF
- early decision on GFP evaluation

Example:

$$\textbf{TargetIdentified} \rightarrow \forall\Diamond\textbf{TargetHit}$$

- **After negation,**

$$\textbf{TargetIdentified} \wedge \exists\Box \neg \textbf{TargetHit}$$

- **Can quit evaluation iff**

the intersection is already empty!!!

134

Linear hybrid automata

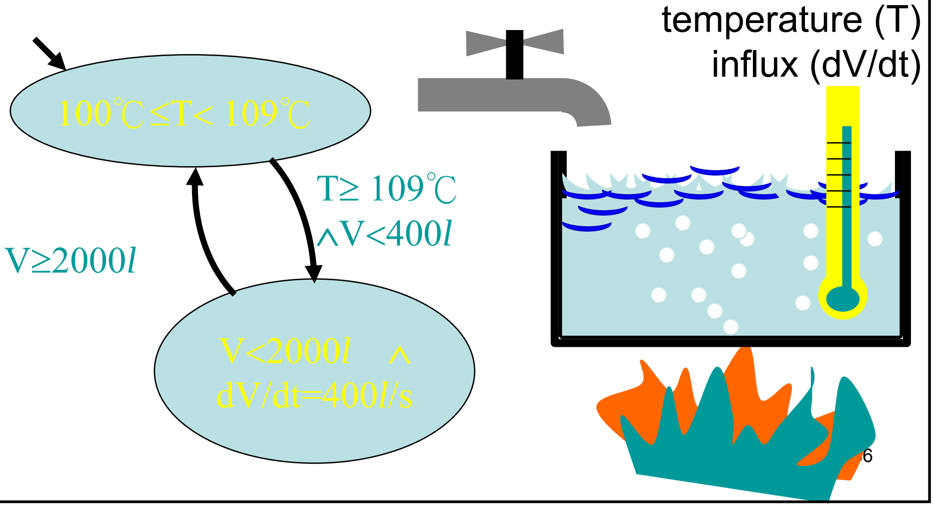
FSA + continuous variables

- **Continuous variables: rationals, reals**
  - temperature, volume, velocity, distance, voltage, …
- **Continuous variables + their derivatives for system state descriptions/transitions**
  - $T > 100℃ \wedge V < 400l \wedge dV/dt < 10l/s$
  - $dT/dt = 6℃/s \wedge V < 400l$

135

Linear hybrid automata - example

A boiler control system with
water volume (V)
temperature (T)
influx (dV/dt)



6

Linear hybrid automata - verification

- **linear state-predicates, e.g.,**

  $6x + y + 5 < 2z + 12$

  and their Boolean combinations
  - *reachability problem is undecidable.*
- **In timed automata with constraints like $5 < y$, $x < 5$**

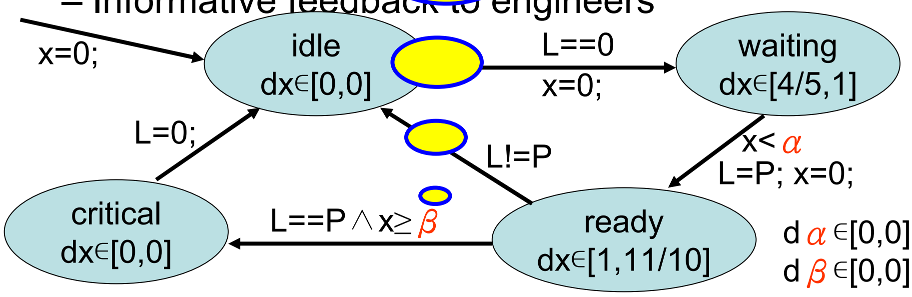
  model-checking problem is in PSPACE.

137

Linear hybrid automata - p alysis

- **Expre tems**
- **Genera ric analysis**
  - Informative feedback to engineers

What values of α and β make the system safe ?



138

Linear hybrid automata

- parametric safety analysis (PSA)

Given an LHA A and a safety predicate η ,

``What is the constraint on parameters that make A safe w.r.t. η ?''

i.e., η is satisfied in all states along all computations of A.

- Decision support
- Informative feedback to engineers
- An undecidable problem

139

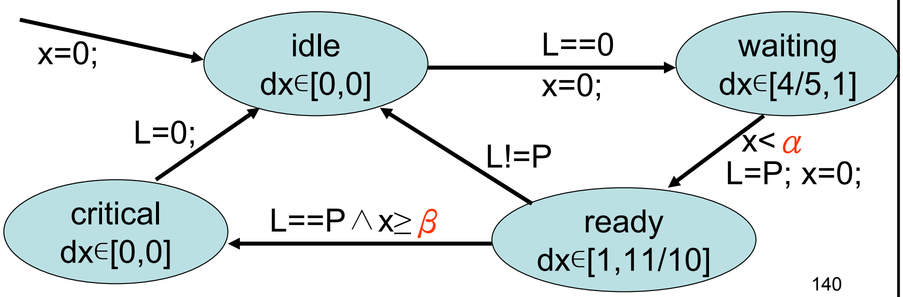
Linear hybrid automata

- parametric safety analysis (PSA)

Given an LHA A and a safety predicate η ,

"What is the constraint on parameters that make A safe w.r.t. η ?''

Safety solution: **$-\alpha<0 \wedge -11\alpha+8\beta<0$**



140

Linear hybrid automata
- How to get rid of ∃δ ?

Example:

x-3y<-5Ù 4x+2y <9, dx/dt ∊ [1,3], dy/dt ∊[-10,-9],

what is the weakest precondition of time-progress ?

Techniques:

present copies of variables are *x', y'*

future copies (after δ time units) are *x, y*

Solution:

∃x' ∃y' ∃δ (x'-3y'<-5∧ 4x'+2y' <9

∧ 0≤δ ∧ δ≤ *x-x'*≤3δ ∧ -10δ ≤ *y-y'* ≤-9δ

)

141

Linear hybrid automata
- related works

The technology for LHA parametric analysis

- **Non-algorithmic in general**
- **State-space analysis te**
  - Convex polyhed
  - Frames
  - Zones for time
    - DBM (Differe
    - DDD, CDD, CRD (D

Can LHA analysis benefit from BDD-technology ?

142

Linear hybrid automata - symbolic verification

Convex polyhedra:

Given system variables $x_1, x_2, \ldots, x_n$,

$$\bigwedge_k a_{1k} x_1 + a_{2k} x_2 + \ldots + a_{nk} x_n \sim c_k$$

- $a_{1k}, a_{2k}, a_{nk}, c_k \in Z$
- $\sim$ is $<, \leq, =, \geq, >$

Example: $11\alpha - 8\beta < 0 \wedge 4\alpha - 5x_1 + 9x_2 < 0$

143

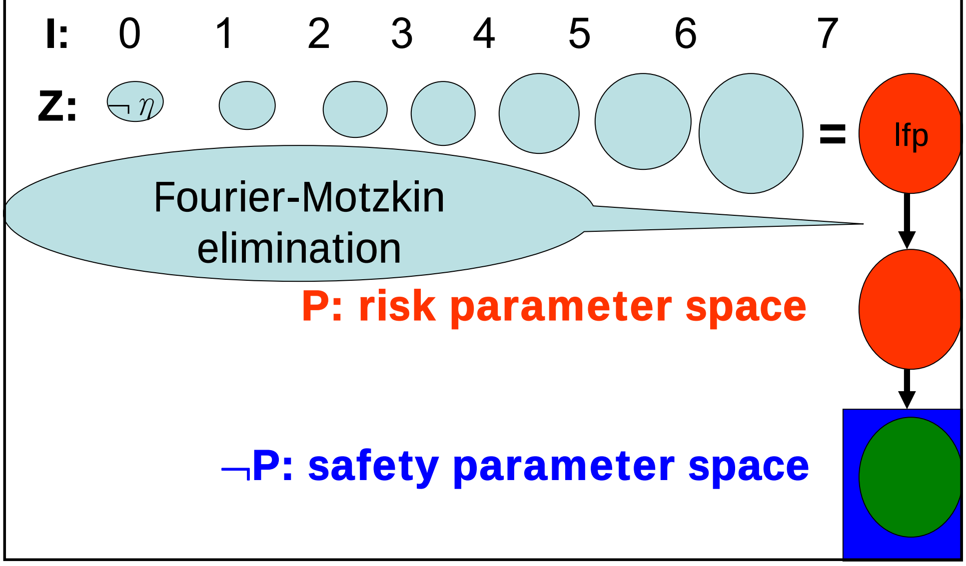
Parametric safety analysis

$\neg \exists (X\text{-}H)$
$(I \wedge \text{lfp } Z.(\text{time}(\neg \eta, q_f) \vee \bigvee_{e=(q,q') \in E} \text{time}(\text{xtion}(Z,e),q)))$

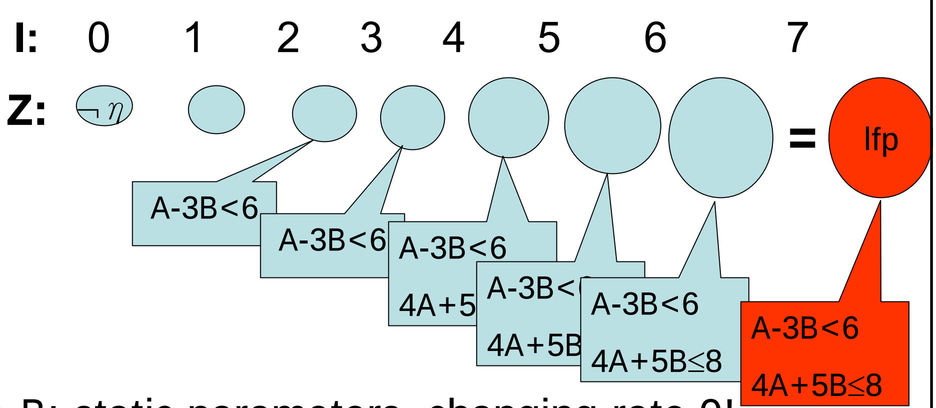
- **H: the set of static parameters**
- η **: safety predicate (in location q_f)**
- **I: initial predicate**
- **time(η',q): WPC through time passage**
- **xtion(η',e): WPC through discrete transition e**

144

Animation of parametric safety analysis



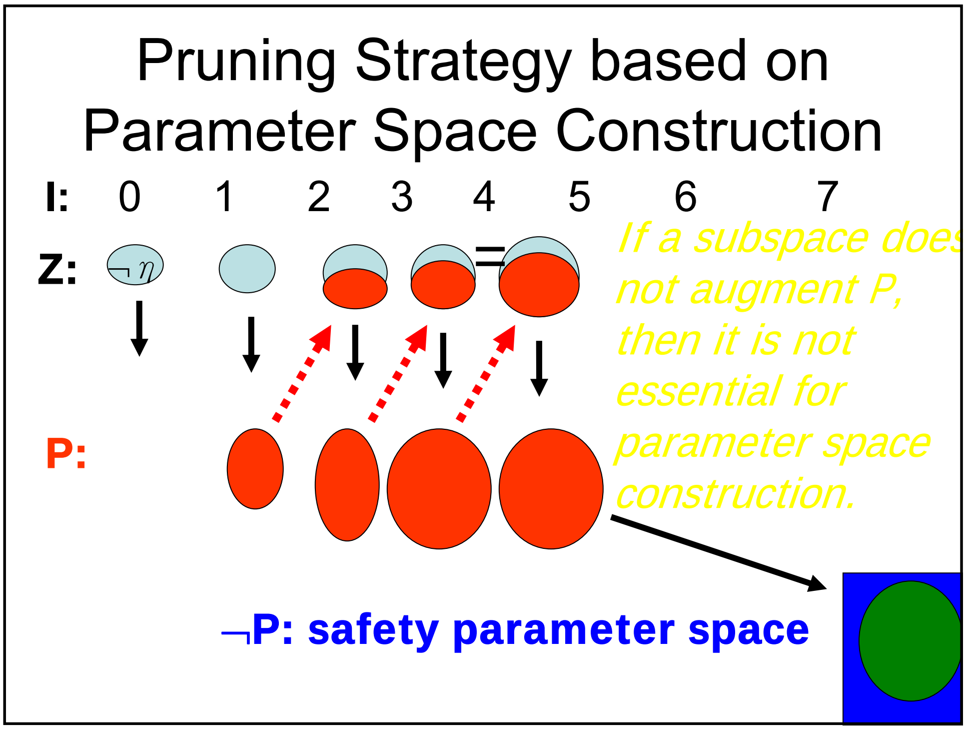
Observation for speedup

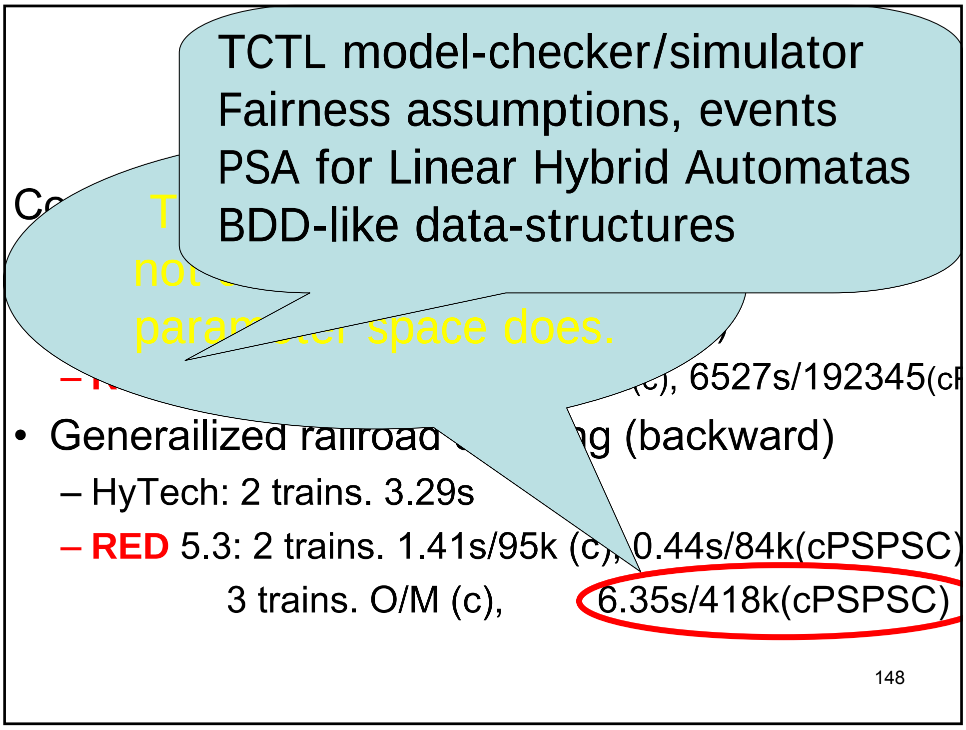


A,B: static parameters, changing rate 0!

Static parameter constraints are stable!

146


Pruning Strategy based on Parameter Space Construction
I: 0 1 2 3 4 5 6 7
Z:
¬η
P:
If a subspace does not augment P, then it is not essential for parameter space construction.
¬P: safety parameter space


TCTL model-checker/simulator
Fairness assumptions, events
PSA for Linear Hybrid Automatas
BDD-like data-structures
parameter space does.
(c), 6527s/192345
• Generailized railroad
ng (backward)
– HyTech: 2 trains. 3.29s
– RED 5.3: 2 trains. 1.41s/95k (c), 0.44s/84k(cPSPSC)
3 trains. O/M (c), 6.35s/418k(cPSPSC)
148

Experiment

When PSPSC does not perform, it charges a constant ration overhead.

– …7k (m), 0.66s/105k(cPS

6 senders … (m), 40.58s/3513k(…)

- CSMA/CD (forward)
  - HyTech: 4 proc. 68.75s
  - **RED** 5.3: 4proc. 17.49s/378k (c), 27.03s/378k(cPSPS
    6proc. 3123s/11525k (c), 4163s/11552k(cPSP

149

Experiment

Comparison with HyTech 2.4.5

- Fischer’s (backward)
  - HyTech: 4 proc. 28.04s
  - **RED** 5.3: 4 proc. 12.38s/215k (m), 5.14s/163k(cPSPS
    6 proc. 1485s/4000k (m), 168.6s/1170k(…)
- Fischer’s (forward)
  - HyTech: 3 proc. 37.89s
  - **RED** 5.3: 3proc. 19.18s/654k (m), 5.59s/538k(cPSPSC

150

Experiment

Comparison with HyTech 2.4.5
- Nuclear reactor (backward)
  - HyTech: 6 rods. 647.8s
  - **RED** 5.3: 6 trains. 839.3s/8191k (m)
    461.8s/6941k(cPSPSC)

151

Experiment

Comparison with TReX 1.3

Parameters,
Clocks,
Lossy channels

- Fischer's (forward)
  - TReX: 2 proc. 1.12s
  - **RED** 5.3: 4 proc. 197s/2714k (m),
    5 proc. 752.7s/5254k(cPSPSC)
- Fischer's (backward)
  - TReX: 2 proc. 8.96s
  - **RED** 5.3: 6 proc. 567.1s/4341k (m),
    170.3s/2798k(cPSPSC)

152

Experiment

Comparison with TReX 1.3

- Nuclear reactor (backward)
  - TReX: 2 rods. O/M
  - **RED** 5.3: 6 rods. 839.3s/8191k (m),
    461.8s/6941k(cPSPSC)

153