REDLIB Tutorial, 16 Dec. 2011

Model-Checking Library Support for Dense-time Systems with Decision Diagrams

Farn Wang
Dept. of Electrical Engineering
National Taiwan University

$4 billion development effort

> 50% system integration & validation cost

2,500,000+1,500,000 lines of codes (most in Ada)

Plan of Presentation

- **Theoretical background**
- Clock-restriction diagrams (CRD) for timed systems
- Preprocessing of events for efficient model-checking
- Other features of RED technology
- REDLIB
  - APIs
  - examples
- Future research issues

$\mathbf{A=\langle Q, X, I, \mu, E, \tau, \pi\rangle}$

Q={monitor， hit}

$X=\{x,z\}$

I=monitor

μ(monitor): $x \leq 500 \wedge z \leq 50$

μ(hit): *true*

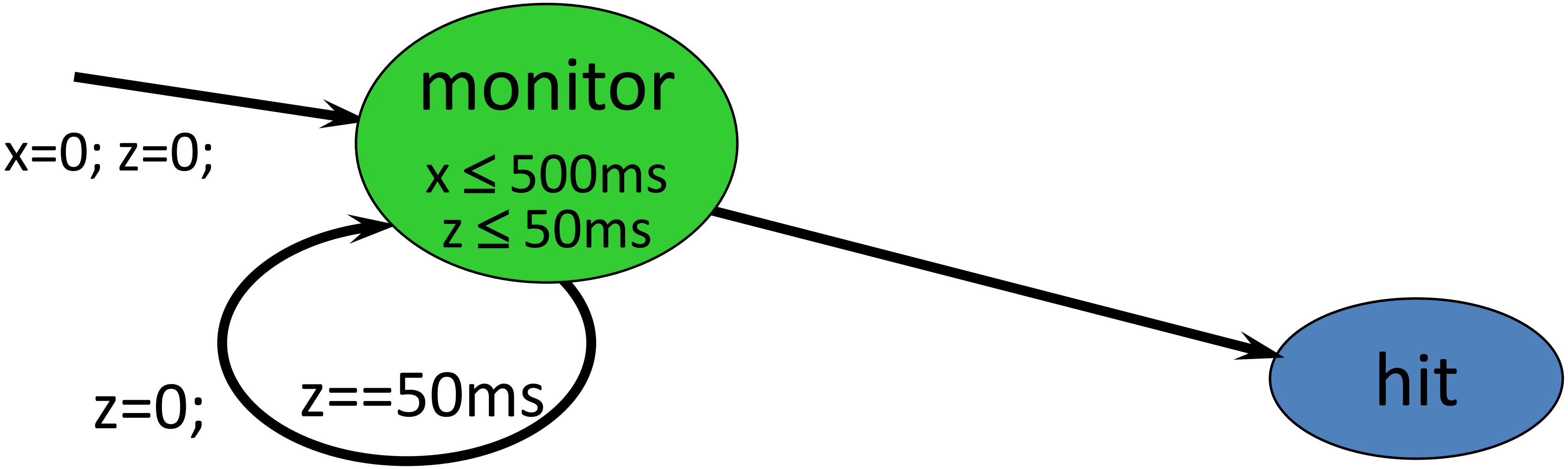
E={(monitor,monitor),(monitor,hit)}

τ(monitor ,monitor): z =50

τ(monitor, hit): *true*

π(monitor ,monitor): {z}

π(monitor, hit): {}



TCTL (Timed Computation-Tree Logic)

$$\phi ::= q \mid x \leq c \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid x.\phi \mid \exists \phi_1 \mathrm{U} \phi_2 \mid \exists \Box \phi$$

Example: *It is possible that I will get my salary at the 7th day.*

$$day.\exists\Diamond(day=7 \wedge \mathrm{salary})$$

Example：*No matter what, I will be married in 10 years.*

$$year.\forall\Diamond(year < 10 \wedge \mathrm{married})$$

TCTL

Example: *After you are married, you can remain happy in five days.*

$$\forall\Box(married \rightarrow day.\exists\Box(day<5\rightarrow happy))$$

Example: *After you are married, you will remain happy in five days.*

$$\forall\Box(married \rightarrow day.\forall\Box(day<5\rightarrow happy))$$

TCTLF (TCTL with fairness)

Example: *If I will be married for infinitely many times, I will always be happy.*

$$\forall^{married}\Box happy$$

Example: *If process 0 and 2 each infinitely many opportunities to execute, then boot completes someday.*

$$\forall^{turn0,\ turn1}\Diamond boot\_complete$$

TCTL verification complexities

- TCTLF model-checking problem against timed automata is PSPACE-complete.
- TCTL satisfiability problem is undecidable.

Alur, Cocoubetis, Dill [IEEE LICS 1990]

Zones

basic objects for manipulation and representation

A zone is a state-space bounded by clock difference constraints.

- $x-x' < d$, $x,x' \in X\cup\{0\}$; $d \in Z\cup\{\infty\}$
- $x-x' \leq d$, $x,x' \in X\cup\{0\}$; $d\in Z$

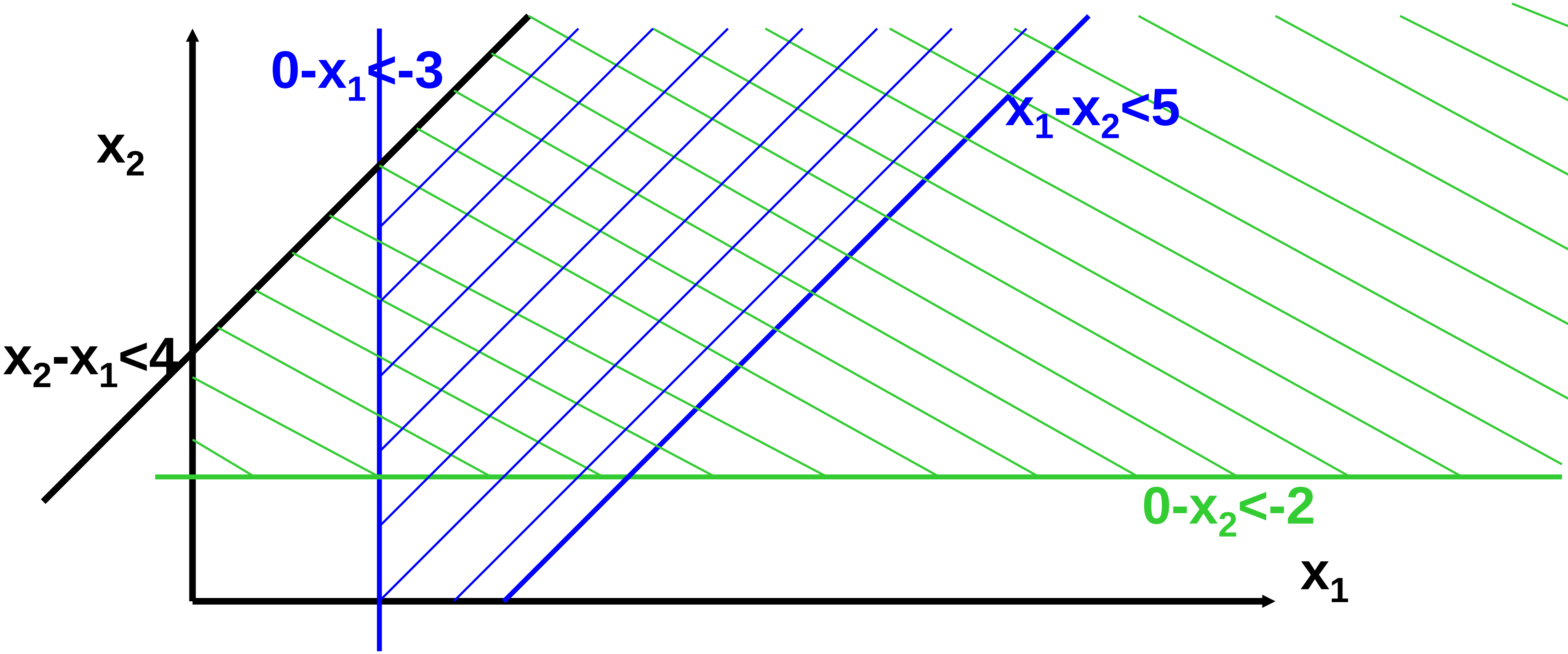
Features:

- convex
- the reachable state-space is a union of zones
- can intersect with one another
- can contain one another → *redundancy*
- non-canonical → *needs normal (or canonical) forms*

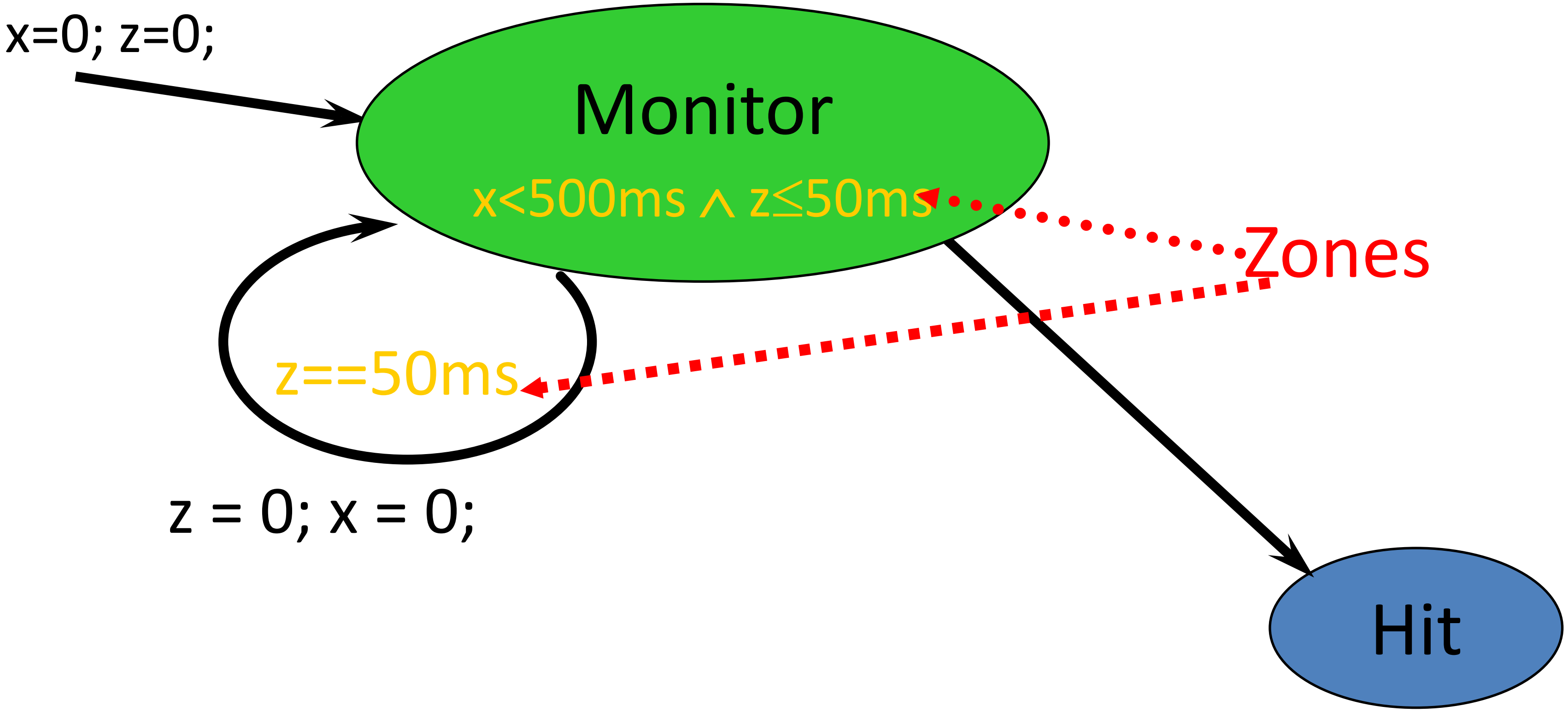
Zones

Two zones

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



Zones in a timed automata



Zones & normalization

Two zones:

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

Normal forms

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

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

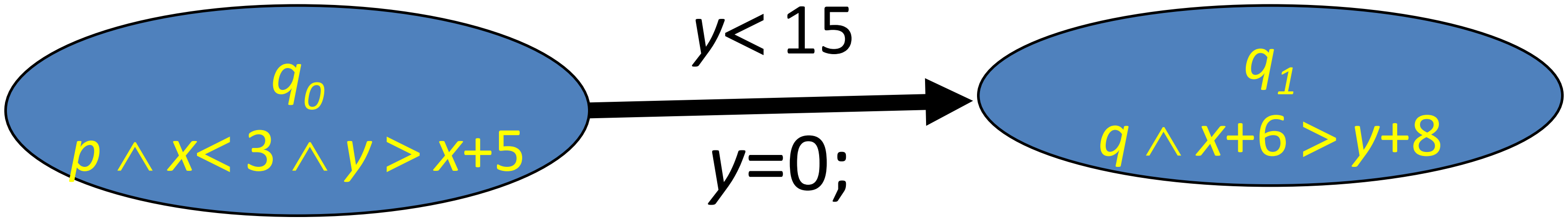
always the most number of constraints

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

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

Symbolic precondition of dense-time state-space

Henzinger, Nicollins, Sifakis, Yovine [IEEE LICS 1992]



What is the weakest precondition in q_0 that

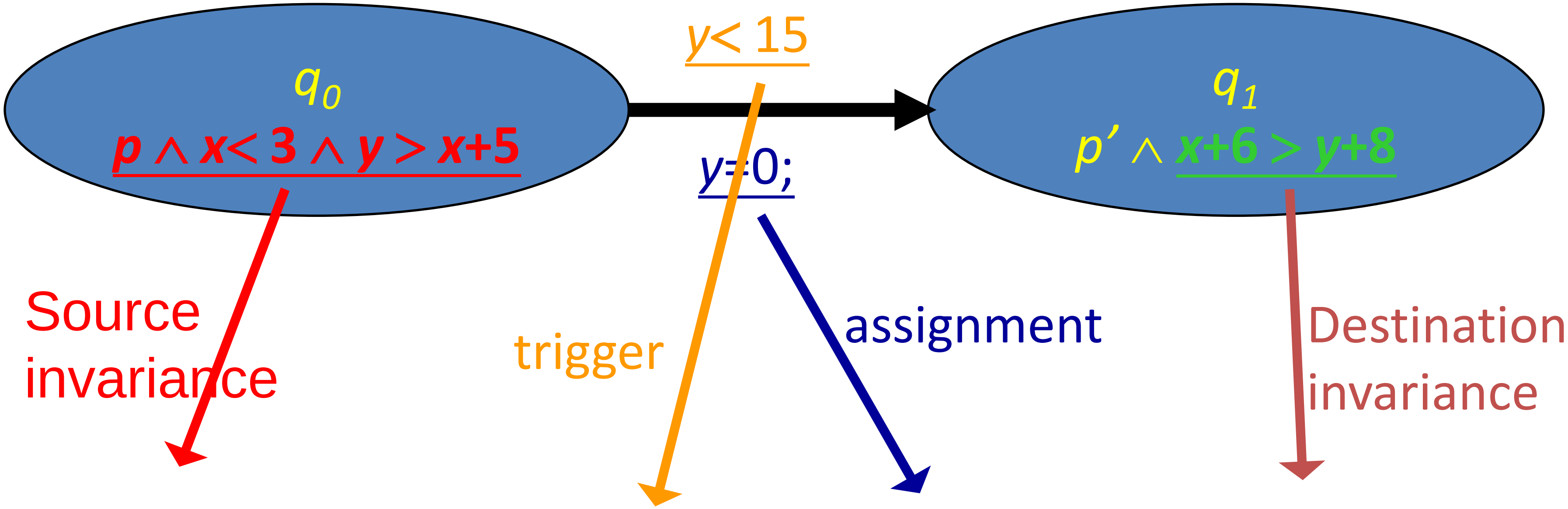
- can transit from q_0 to q_1 ?

  xbck(e,η) : the weakest precondito in η after transition e ?

- after time t ≥ 0 , remains in s_0 ?

  T(φ, η) : the weakest precondition to η through time-progress in φ?

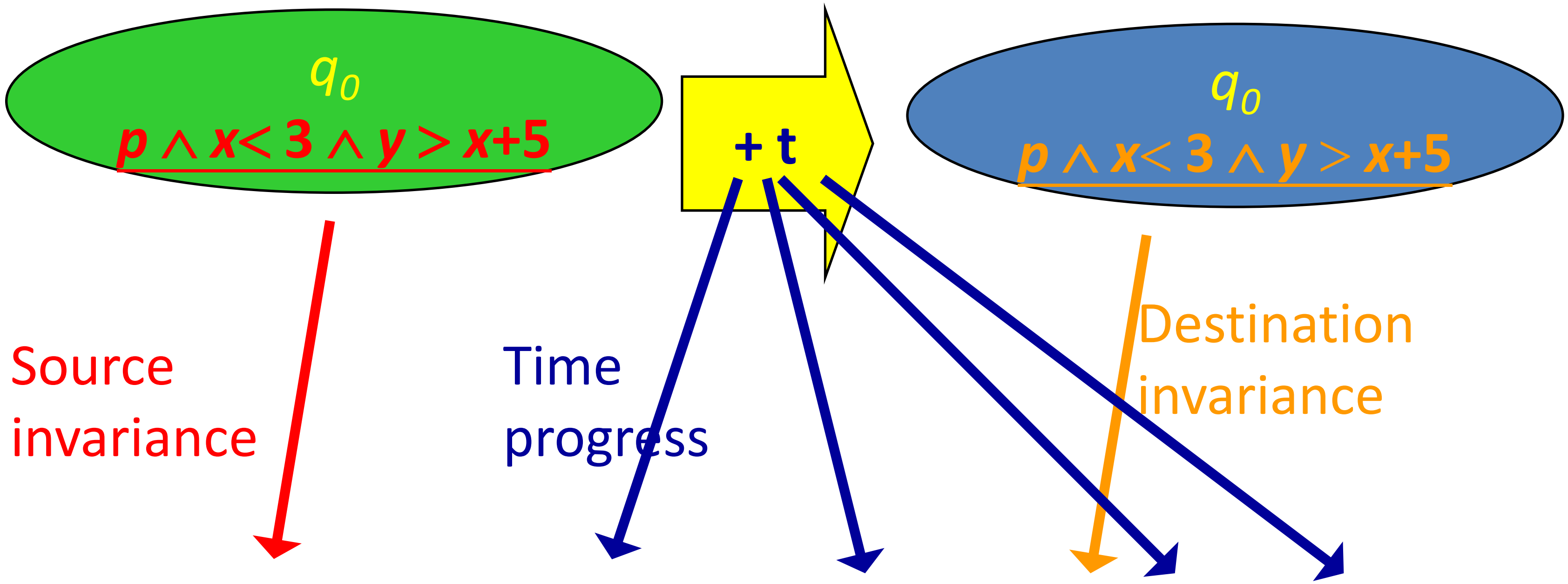
xbck(e,η)
weakest preconditoin to e ?



$$\underline{p \wedge x< 3 \wedge y > x+5} \wedge \underline{y< 15} \wedge \underline{\exists y(y=0} \wedge \underline{x+6 > y+8}\wedge\eta)$$

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

T(φ,η) : the weakest precondition to η through time-progress t ?



q_0

$p \wedge x< 3 \wedge y > x+5$

+ t

q_0

$p \wedge x< 3 \wedge y > x+5$

Source invariance

Time progress

Destination invariance

$$p \wedge x<3 \wedge y>x+5 \wedge \exists t(0\leq t \wedge x+t <3 \wedge y+t>x+t+5)$$

How to get rid of the ∃t ?

Evaluation of existential quantification

$$\exists y(x \leq y \wedge y < z+5) \equiv \quad x < z+5$$

Joseph Fourier, 1824!
(1768-1830)

TCTL Model checking procedures

- Basic procedures

  xbck(e, η)

  - weakest precondition of discrete transitions

  T(φ ,η)

  - backward time-progression

- Backward reachability:

  $\mathrm{rbck}(\varphi,\eta) \equiv \mathrm{lfp}Y.(\eta\vee T(\varphi \wedge \bigvee \mathrm{xbck}(e,Y)))$

least fixpoint operator

TCTLF Model checking

Lemma: given $d \geq 1$,

$A, \nu \models \exists \Box \eta$ iff there is a finite run ρ

- from ν
- of duration $\geq d$
- along ρ every state satisfies η and
- ρ ends at a state satisfying $\exists \Box \eta$

$$\exists \Box \eta \equiv \text{gfp } Y.\ \exists z(\text{rbck}(\eta, Y \wedge z \geq d))$$

greatest fixpoint operator

Plan of Presentation

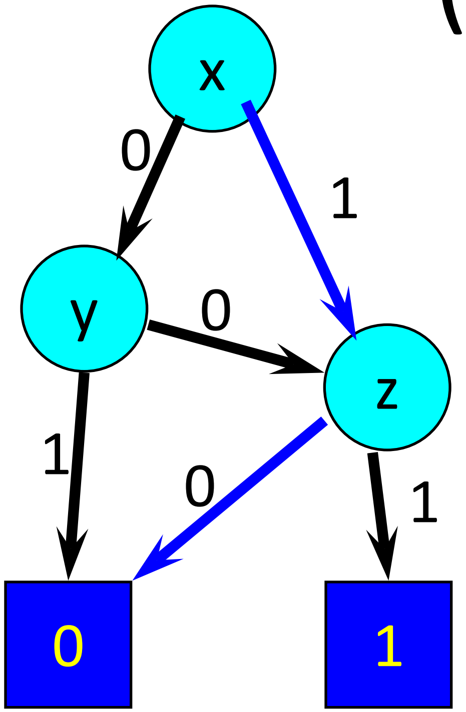
- Theoretical background
- **Clock-restriction diagrams (CRD) for timed systems**
- Preprocessing of events for efficient model-checking
- Other features of RED technology
- REDLIB
  - APIs
  - examples
- Future research issues

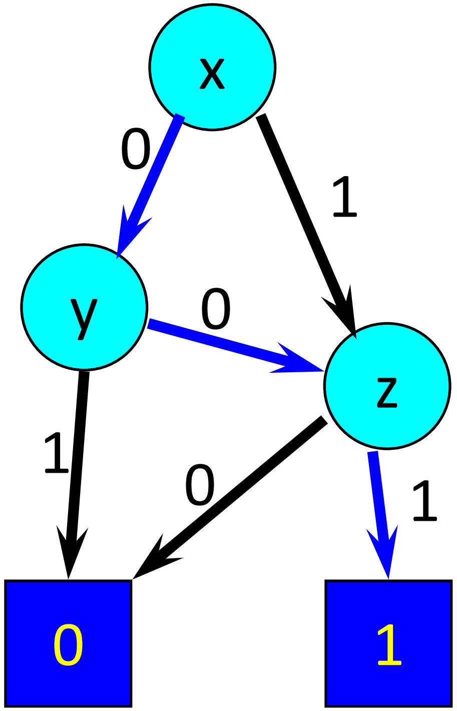
BDD in evaluating function values

Minimal & Canonicity

$$(x \vee \neg y) \wedge z$$

x 1
⇓
y 0
⇓
z 0



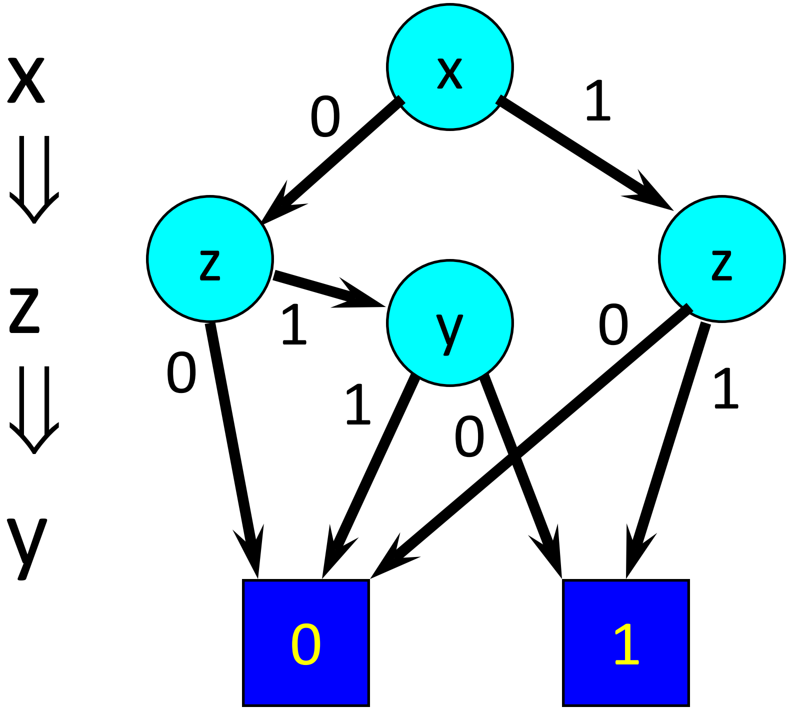


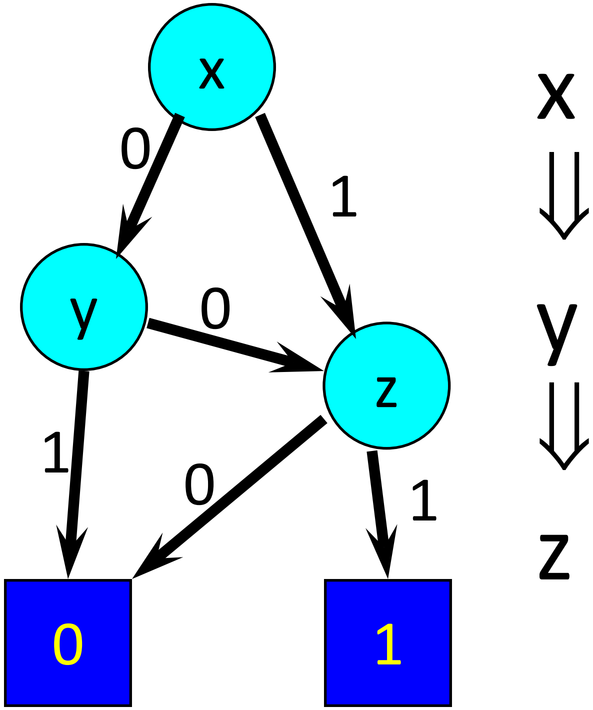
0 x
⇓
0 y
⇓
1 z

BDD

Effects of variable ordering

$$(x \vee \neg y) \wedge z$$





BDD operations

Given 2 BDDs ：B_1、B_2，efficient algorithms for the following:

- $B_1 \wedge B_2$
- $B_1 \vee B_2$
- $\neg B_1$
- $\exists z\ B_1$

R. Bryant [IEEE Trans. Computers, 1986]

Clock-Restriction Diagram (CRD)

Background

Intuition: To repeat the success of BDD in circuit verificaito.

Up to 2000, many proposals for BDD variations for dense-time state-spaces.

- NDD, DDD, CDD, **RED**
- None of them shows advantage over DBM.

Motivation

Find out the reasons and ways for improvements.

CRD

Related work (I)

- BDD: for untimed systems *[Bryant 86; Burch, Clarke, et al, 90]*
- DBM: 2-dimensional matrix for a region. *[Dill 89]*
- NDD: BDD to encode discrete time state-spaces. *[ABKMPR97]*
- CDD: a decision diagram for dense-time state-space membership *[BLPWW99].* Like CRD, except:
  - default value is (- ∞, ∞).
  - a value [c,d] of variable x-x' means c≤x-x'≤d.
- DDD: variable like x-y~ c *[WME92,Balarin96,MLAH99]*
  - ***Exponential number of BDD atoms***
- **RED**: encodes the ordering of the fractional parts of clock readings with a single-bit varible *[Wang 00]*
  - for symmetric systems

CRD

Related work (II)

Difference-Bound Matrix (DBM) *[Dill 89]*

- 2-dimensional matrix for a convex clock value space.
- A zone is represented as a pair (b,z)
  - b is a BDD for the discrete part of the states
  - z is a DBM for clock values
- Can only represent a convex state space.
- To represent a non-convex state space, a linked list of such pairs are used.
- Inefficient in conjunction and complementation.
- Adopted by UPPAAL, Kronos, Prism, …

Clock-Restriction Diagram (CRD)

- A BDD variation
- Recording device for (zone) DBM sets
- Variables of the form x-x’
- Arc values of the form:

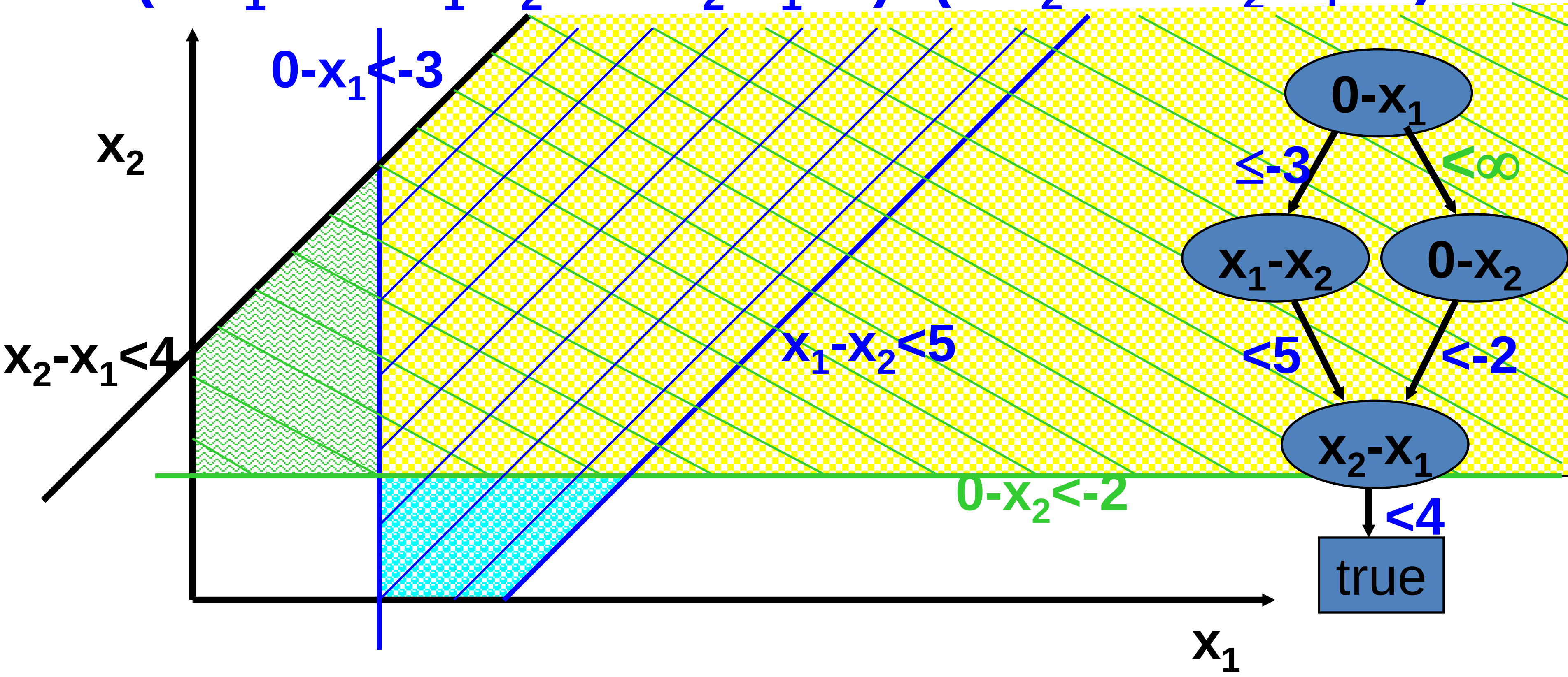
  $(<, d)$, $d \in [-c_A, c_A] \cup \{\infty\}$; or

  $(\leq, d)$, $d \in [-c_A, c_A]$
- Default value on arcs: $(<, \infty)$
  - representing no constraint!

Representation in CRD

Two zones

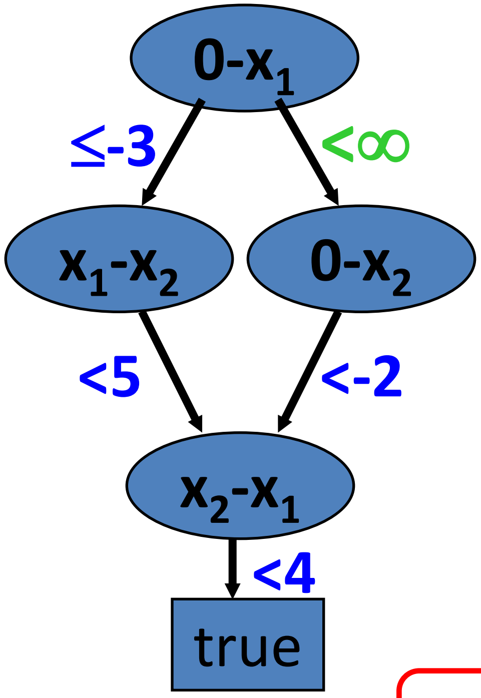
$(0-x_1 \leq -3 \wedge x_1-x_2<5 \wedge x_2-x_1<4) \vee (0-x_2<-2 \wedge x_2-x_1<4)$



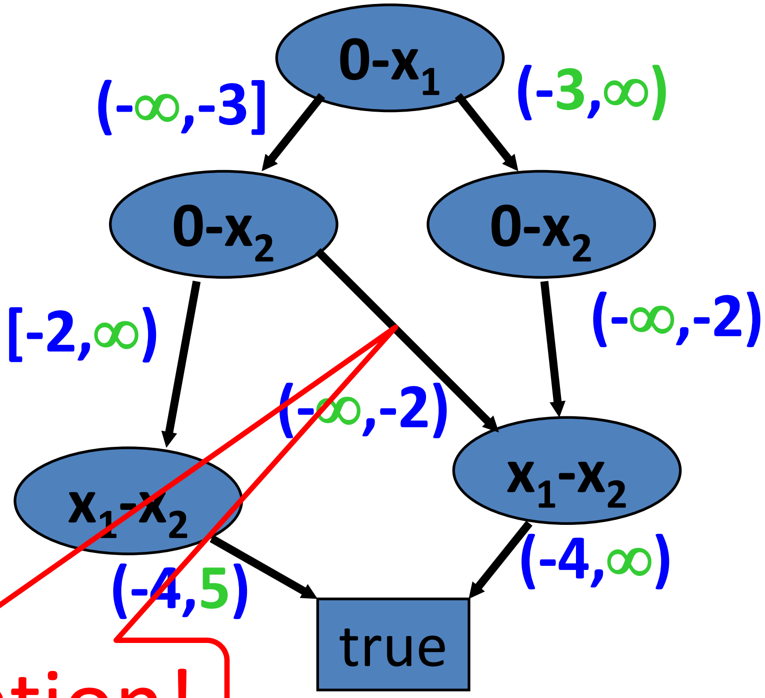
CRD

Example

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



(a) in CRD



Fragmentation!

(b) in CDD

CRD

Characteristics

- Integrated representation for convex and non-convex state spaces
- Efficient operation for union and intersection
- Complementation is a little bit complex.
- No natural canonicity
  - needs normalization

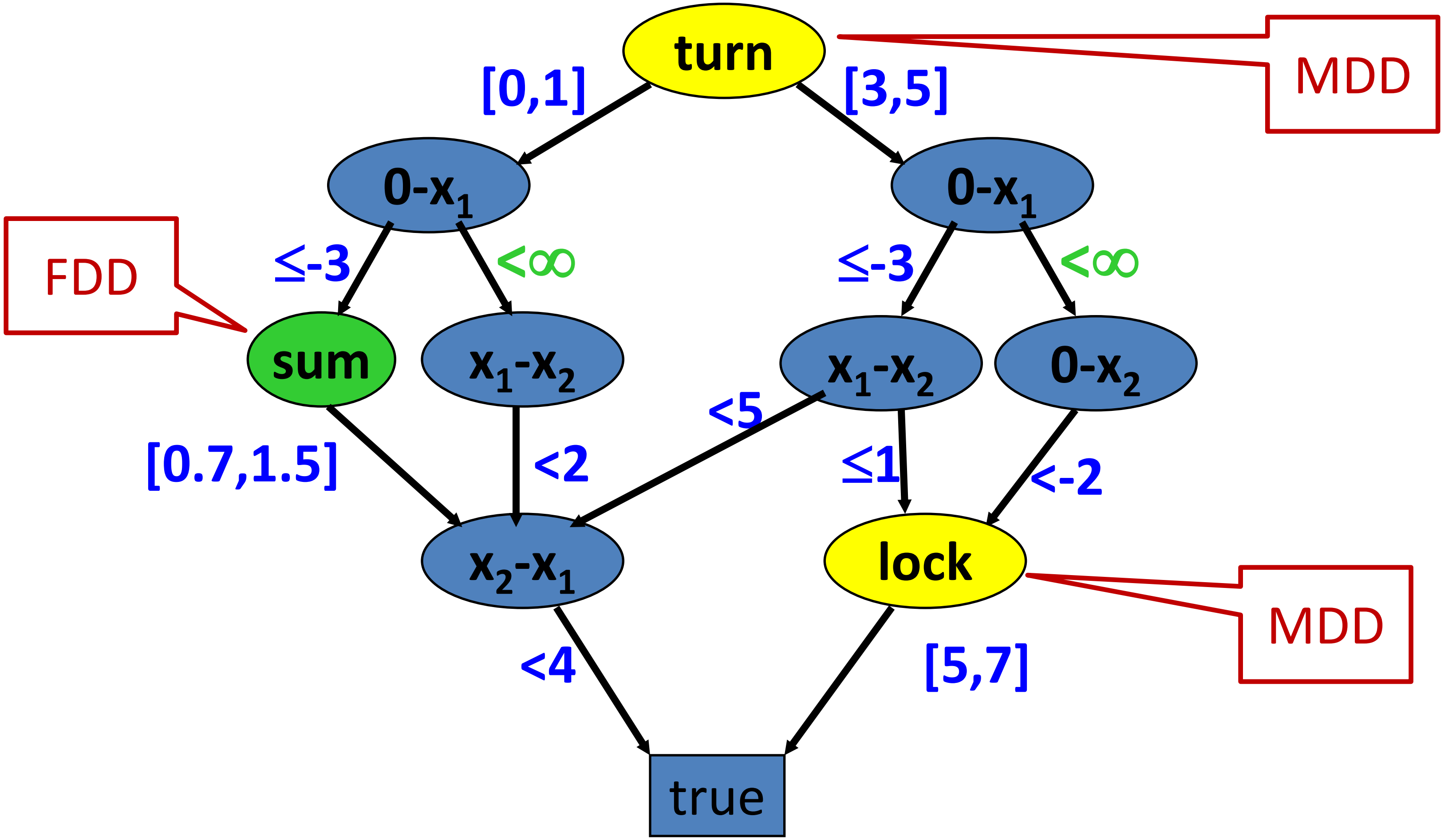
CRD+MDD+FDD

*A winner for **RED** technology.*

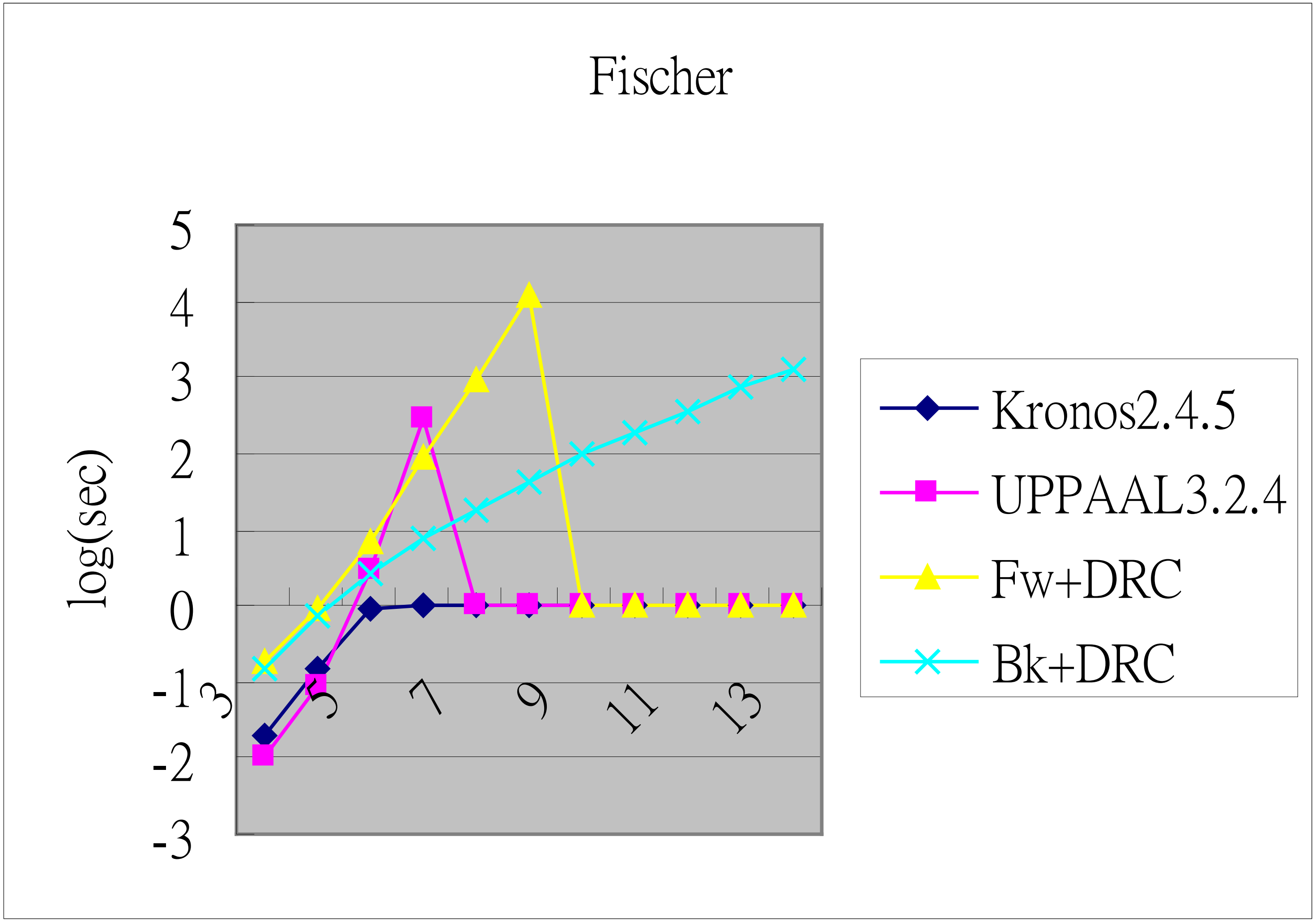
- Integrated representation for convex and non-convex state spaces
- Efficient operation for union and intersection
- Complementation is a little bit complex.
- No natural canonicity
  - needs normalization

CRD+MDD+FDD

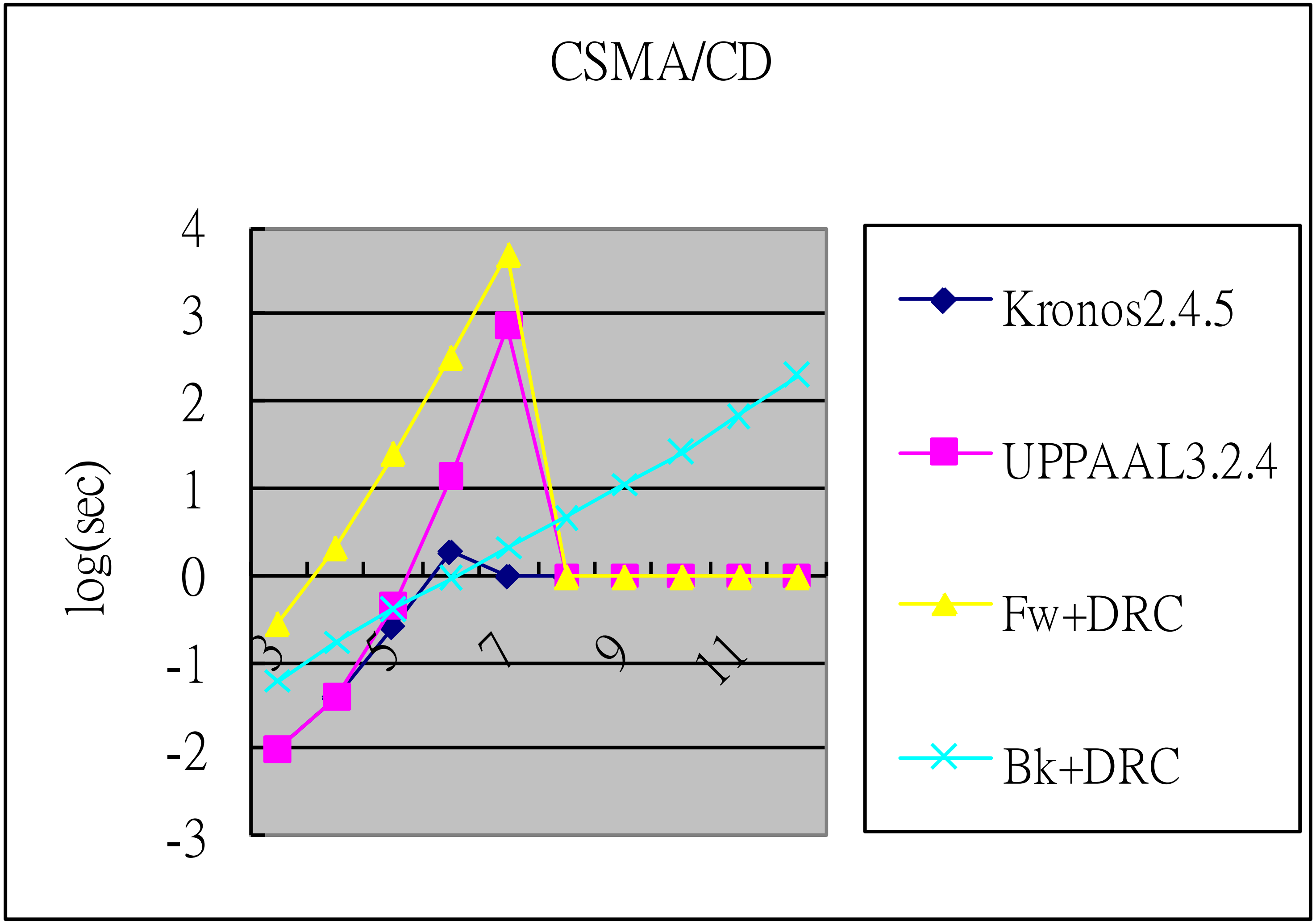
Example



Fischer’s mutual exclusion



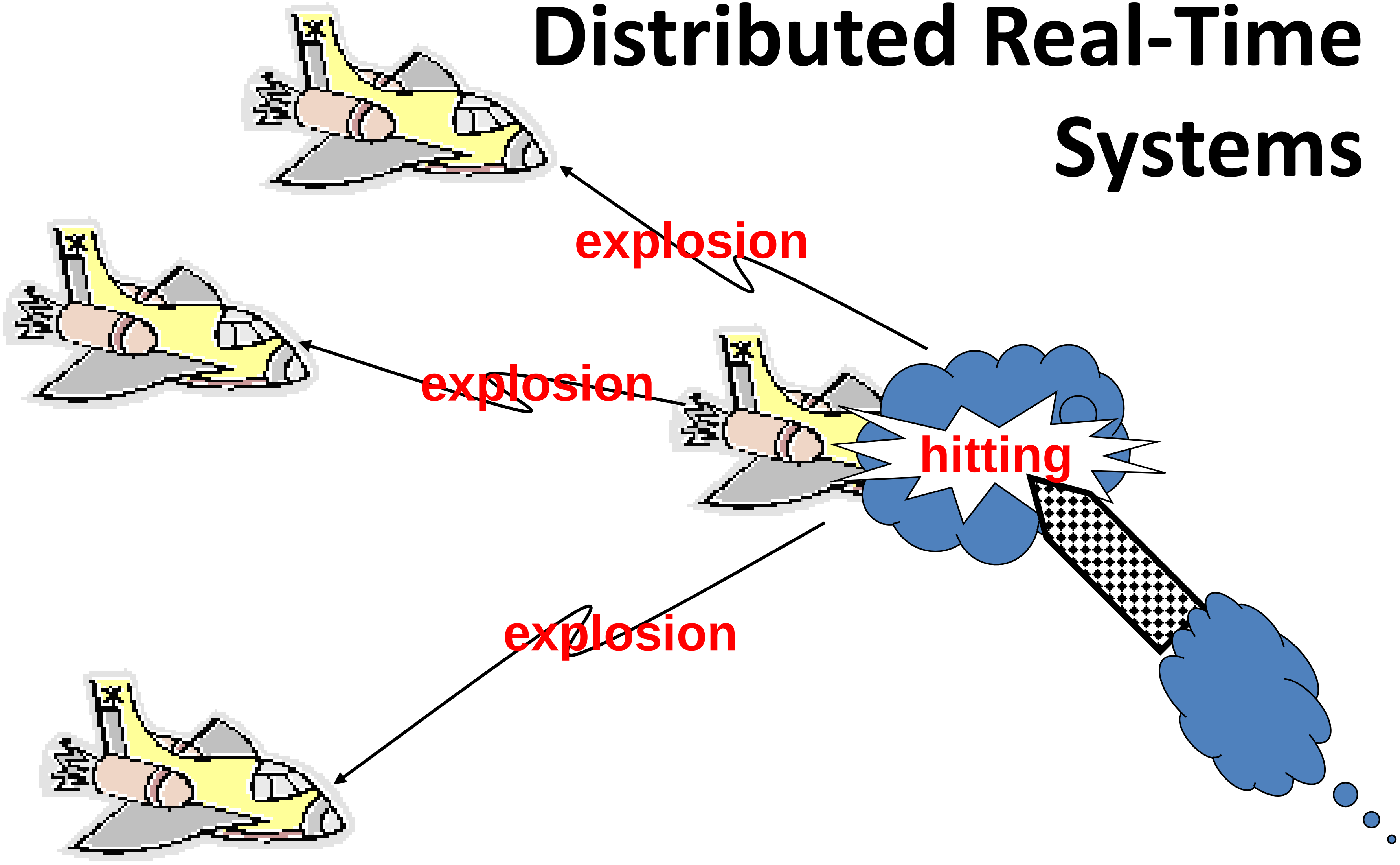
CSMA/CD


CSMA/CD
log(sec)
4
3
2
1
0
-1
-2
-3
3
5
7
9
11
Kronos2.4.5
UPPAAL3.2.4
Fw+DRC
Bk+DRC

Plan of Presentation

- Theoretical background
- Clock-restriction diagrams (CRD) for timed systems
- **Preprocessing of events for efficient model-checking**
- Other features of RED technology
- REDLIB
  - APIs
  - examples
- Future research issues

Complex Synchronizations in Distributed Real-Time Systems

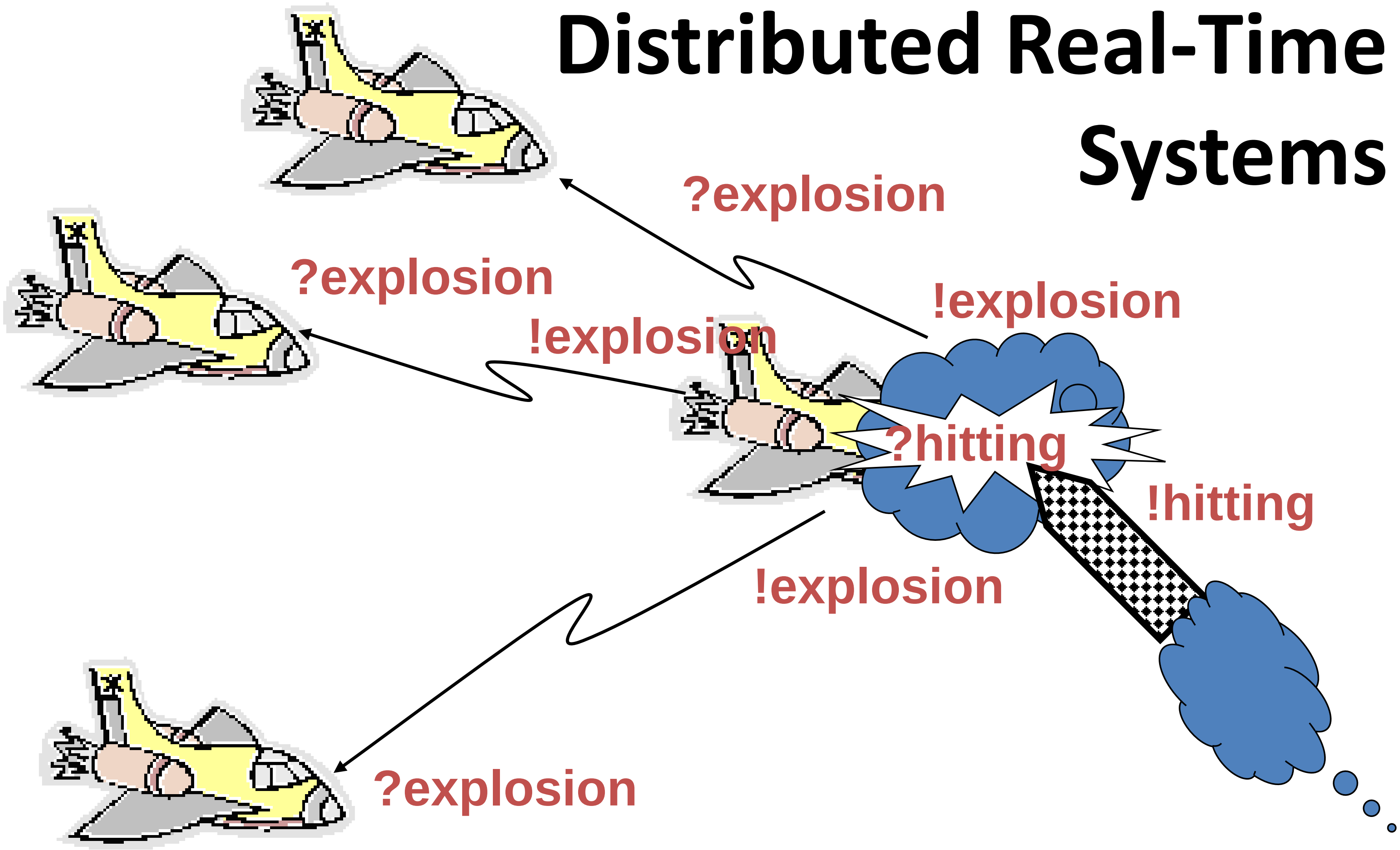


Modeling granularity ?

Did the hit and explosion happen at the same time ?

- ~~No, at the fine level of quantum mechanics~~
  - ~~Extremely fine models~~
  - ~~Extreme confidence in the verification result~~
- **Yes, for the complexity of verification**
  - Intuitive, natural, and acceptable
  - Usable analysis result

Complex Synchronizations in Distributed Real-Time Systems



Complex Synchronizations in the CSP style

- Multiple-party synchronizations constructed through binary synchronizations
  - Global transitions constructed through process transitions
  - For each channel,

    # input event = # output event
  - For interleaving semantics

    Minimality of global transitions
- Modular descriptions and specifications
  - Flexibility in the descriptions of process response variations

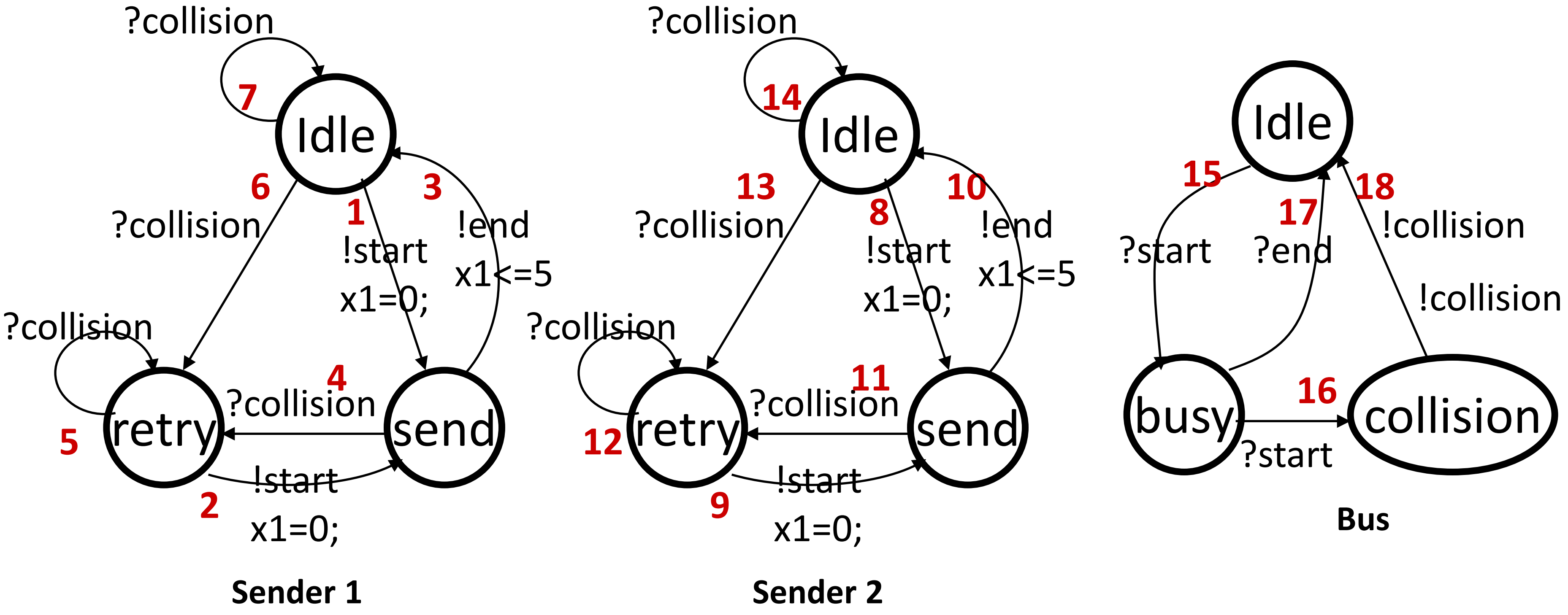
Network of TAs

Communicating timed automaton (CTA)

Legitimate global transitions

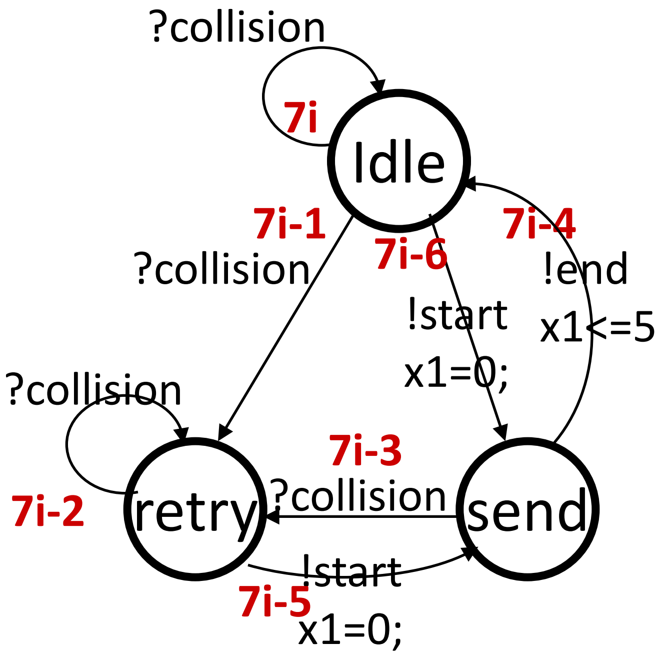
start: (1,15), (1,18), (2,15), (2,18), (8,15), (8,18), (9,15), (9,18)

collision: (4,11,18)

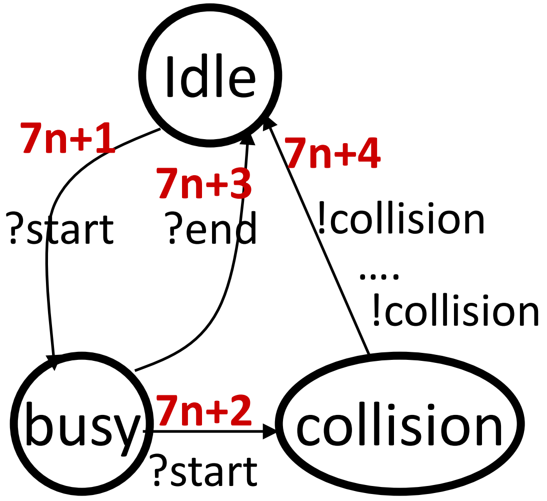


Legitimate global transitions
with *n* senders

When n = 3, collision: (4,11,19,25) (4,11,20,25) (4,11,21,25)
(4,12,18,25) (4,13,18,25) (4,14,18,25)
(5,11,18,25) (6,11,18,25) (7,11,18,25)



Sender i



Bus

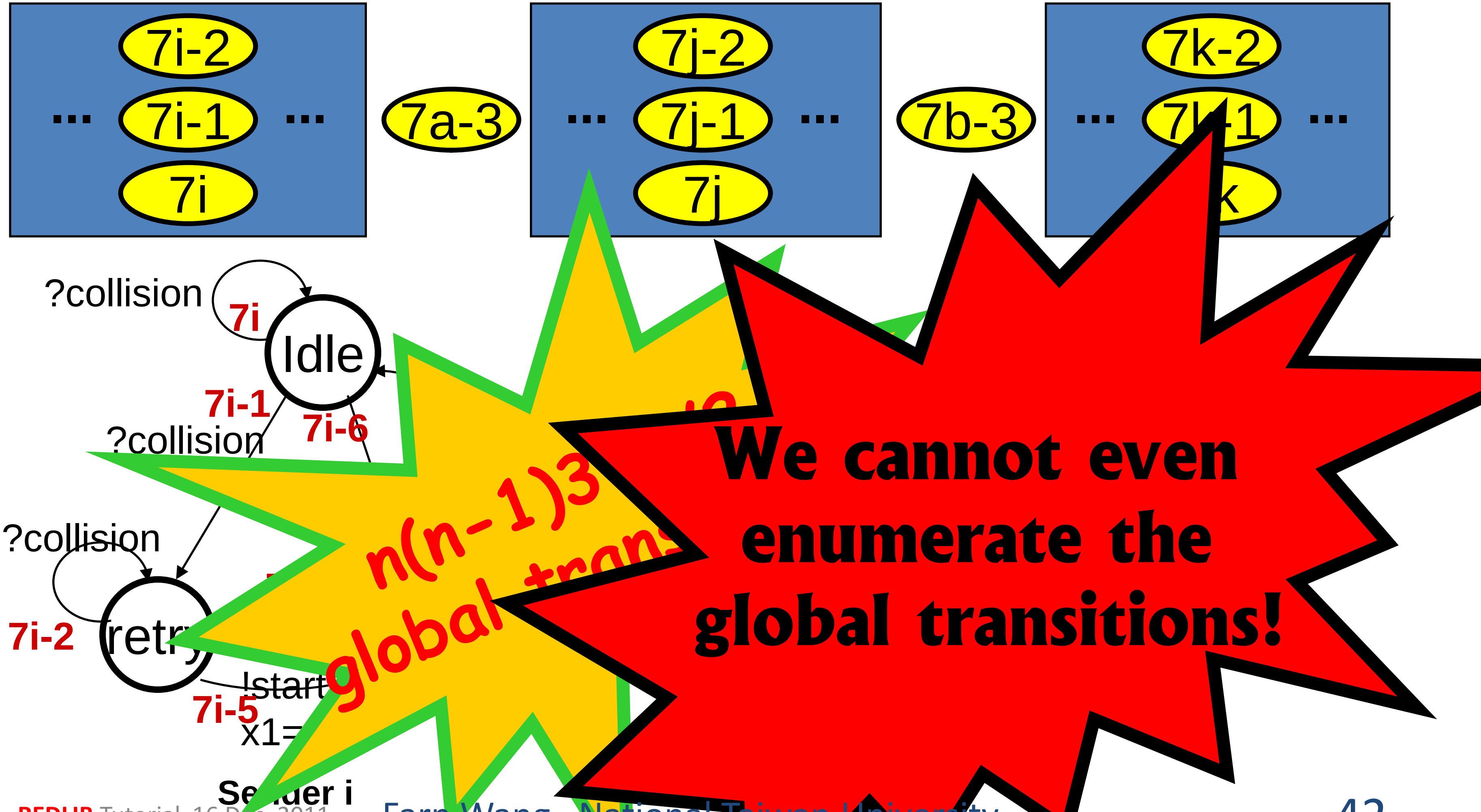
Pre/post condition calculation in the traditional style

```
$\psi$:= false;
for each global transition T {
  for each $1\leq p\leq m$ with $T(p)\neq\perp$ and )=(q,
    $\phi := \eta\wedge (\bigwedge_{x\in\pi p(e)} x=0) \wedge mode_p=q$
    $\phi$ := FM_elim($\phi$ , $\pi_p(e)\cup\{ mode_p\}$) ;
  }
  Add in the triggering conditions
     $\phi$.
  $\psi := \psi\vee \phi$;
}
Return $\psi$;
```

An enumeration of global transitions

Legitimate global transitions

with *n* senders in the traditional style



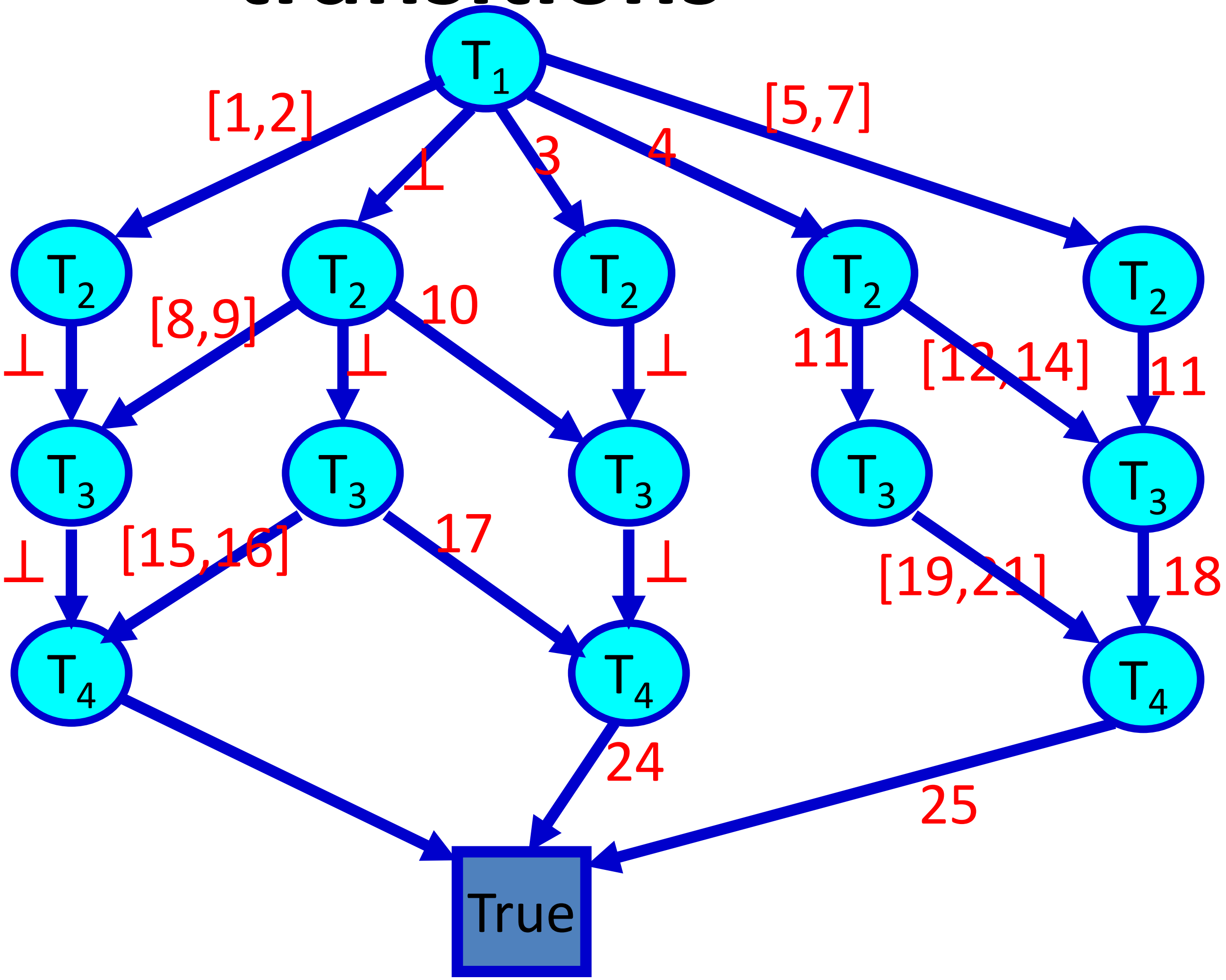
n(n-1)3

global trans

We cannot even enumerate the global transitions!

Efficient representation for global transitions

Γ_M



Xplans_bck(η)

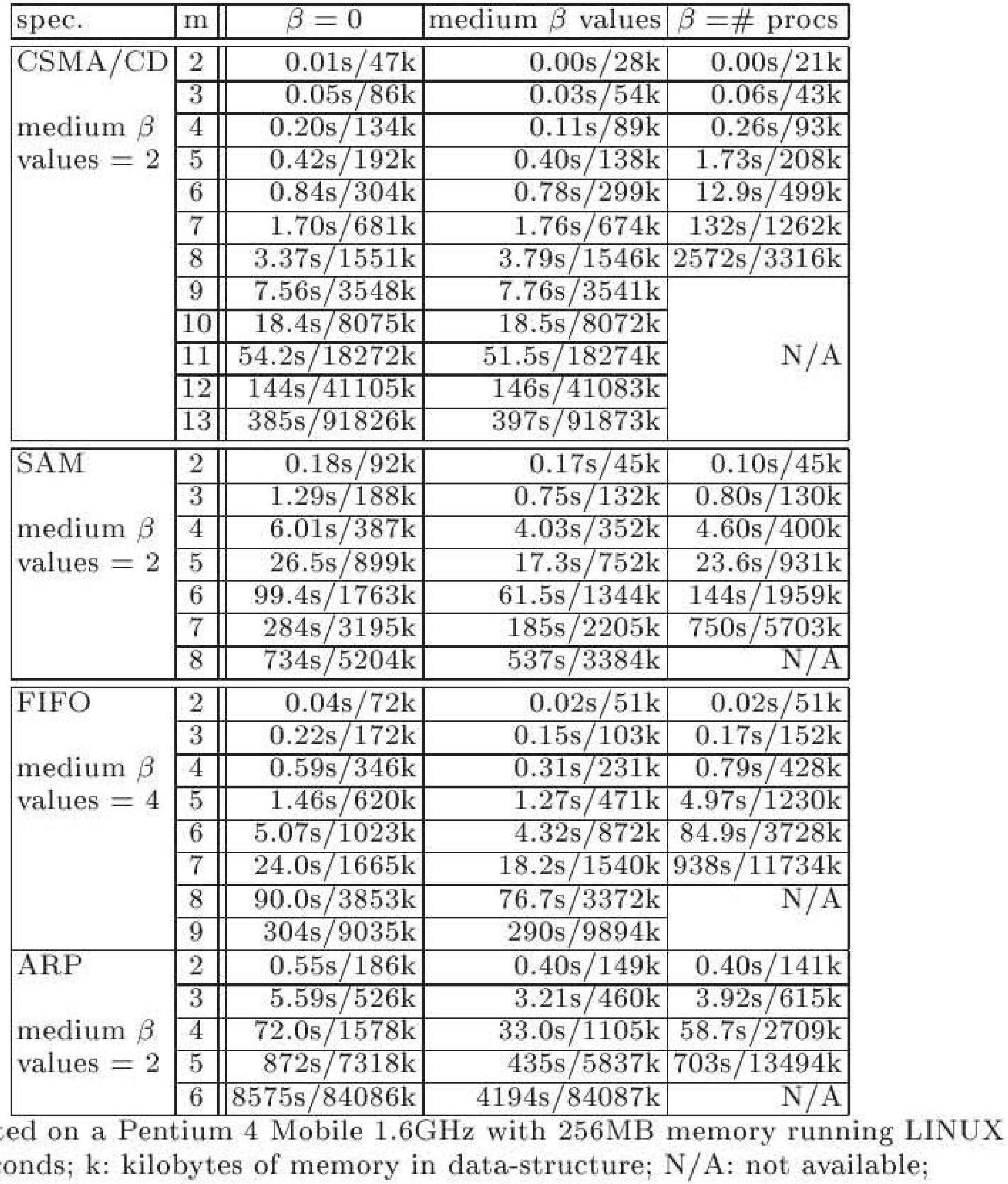
```
Let $\eta := \eta \wedge \Gamma_M$;
for p:=1 to m {
  $\psi := \eta \wedge T_p = \perp$;
  for each e $\in E_p$ with $\gamma(e)=(q,q')$ {
    $\phi := \eta \wedge T_p = e \wedge (\bigwedge_{x \in \pi p(e)} x=0) \wedge mode_p = q'$;
    $\psi := \psi \vee$ FM_elim($\phi$ , $\pi_p(e) \cup \{ mode_p \}$) ;
  }
  $\eta := \psi$;
}
Add in all the triggering conditions of the participating
    process transitions to $\eta$.
return FM_elim($\eta$, $\{T_1,\ldots,T_m\}$);
```

An observation in the experiments

- Simple synchronizations
  - 2 or 3 processes involved
  - Perform well with the traditional style
- Complex synchronizations
  - Perform well with the new style
- Strategy, β : a parameter

  Synchronization
  - < β → traditional style
  - ≥ β → new style

Performance data



spec.	m	$\beta = 0$	medium β values	$\beta =$# procs
CSMA/CD	2	0.01s/47k	0.00s/28k	0.00s/21k
	3	0.05s/86k	0.03s/54k	0.06s/43k
medium β	4	0.20s/134k	0.11s/89k	0.26s/93k
values = 2	5	0.42s/192k	0.40s/138k	1.73s/208k
	6	0.84s/304k	0.78s/299k	12.9s/499k
	7	1.70s/681k	1.76s/674k	132s/1262k
	8	3.37s/1551k	3.79s/1546k	2572s/3316k
	9	7.56s/3548k	7.76s/3541k	N/A
	10	18.4s/8075k	18.5s/8072k	
	11	54.2s/18272k	51.5s/18274k	
	12	144s/41105k	146s/41083k	
	13	385s/91826k	397s/91873k	
SAM	2	0.18s/92k	0.17s/45k	0.10s/45k
	3	1.29s/188k	0.75s/132k	0.80s/130k
medium β	4	6.01s/387k	4.03s/352k	4.60s/400k
values = 2	5	26.5s/899k	17.3s/752k	23.6s/931k
	6	99.4s/1763k	61.5s/1344k	144s/1959k
	7	284s/3195k	185s/2205k	750s/5703k
	8	734s/5204k	537s/3384k	N/A
FIFO	2	0.04s/72k	0.02s/51k	0.02s/51k
	3	0.22s/172k	0.15s/103k	0.17s/152k
medium β	4	0.59s/346k	0.31s/231k	0.79s/428k
values = 4	5	1.46s/620k	1.27s/471k	4.97s/1230k
	6	5.07s/1023k	4.32s/872k	84.9s/3728k
	7	24.0s/1665k	18.2s/1540k	938s/11734k
	8	90.0s/3853k	76.7s/3372k	N/A
	9	304s/9035k	290s/9894k	
ARP	2	0.55s/186k	0.40s/149k	0.40s/141k
	3	5.59s/526k	3.21s/460k	3.92s/615k
medium β	4	72.0s/1578k	33.0s/1105k	58.7s/2709k
values = 2	5	872s/7318k	435s/5837k	703s/13494k
	6	8575s/84086k	4194s/84087k	N/A

data collected on a Pentium 4 Mobile 1.6GHz with 256MB memory running LINUX
s: seconds; k: kilobytes of memory in data-structure; N/A: not available;

Plan of Presentation

- Theoretical background
- Clock-restriction diagrams (CRD) for timed systems
- Preprocessing of events for efficient model-checking
- **Other features of RED technology**
- REDLIB
  - APIs
  - examples
- Future research issues

Other features of RED technology

- Full TCTLF model-checking
  - also with event extensions in TCTLF
  - early termination techniques in greatest fixpoint evaluation.
- Fair simulation for communicating timed automatas
  - model vs. spec TAs in an environment
- Time progress evaluation speed-up
  - [ATVA 2008, RTSS 2008, RTSJ 2011]
  - TCXTL for fast time progress evaluation
  - fast approximation in timed inevitabilities
- Abstraction based on game concepts

REDLIB

library for shared CRD+MDD+FDD

- Support modeling C data-structrures
  - multi-dimensional arrays.
  - structures
  - dynamic memory allocation
    - must first declare the size of memory
  - address arithmetics and indirection
- For model/simulation-checking,
  - a forward untimed reachability analysis is performed.
  - Then a backward timed model/simulation checking is performed
    - to avoid blow-up of the backward state-space.

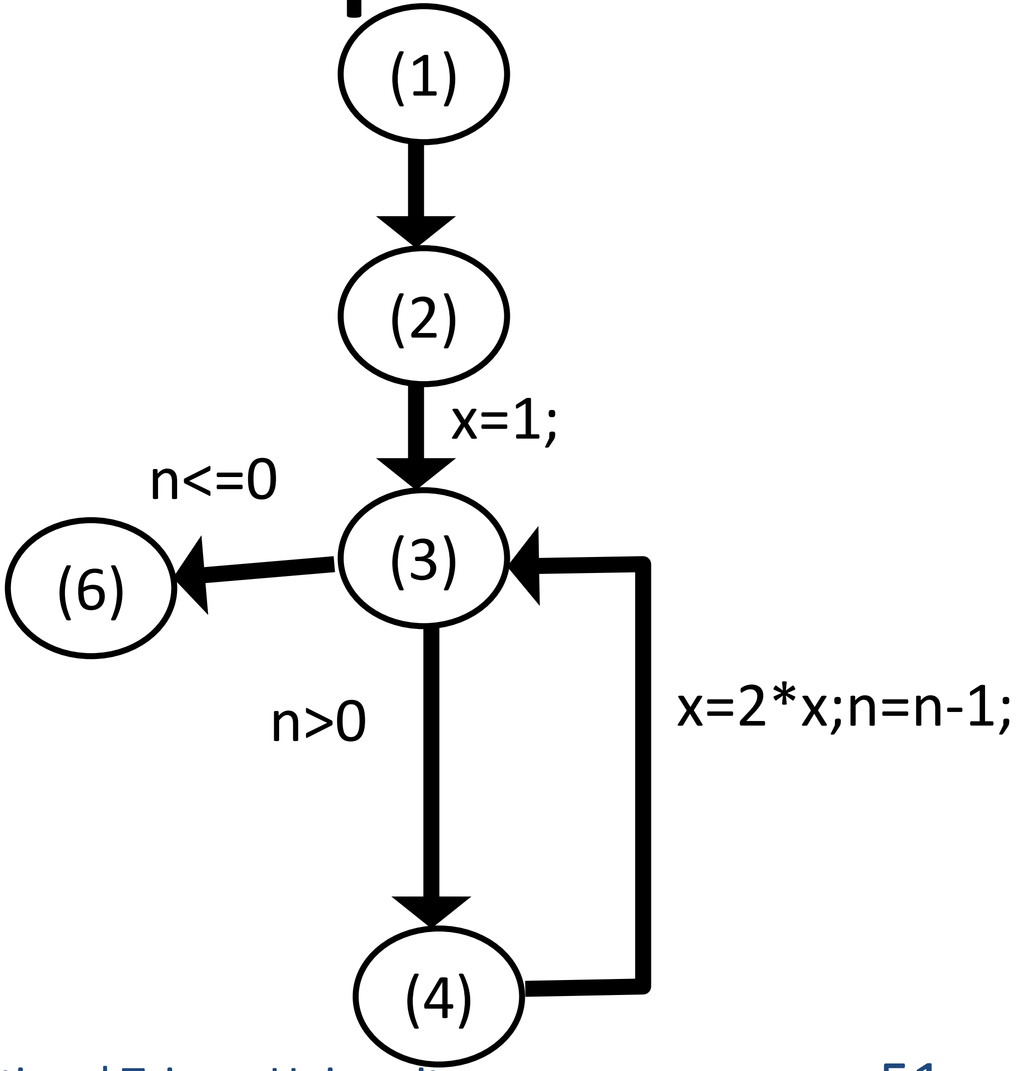
REDLIB

API

- Input
  - parsing, table construction
- Low-level
  - true, false, $\vee$, $\wedge$, $\neg$, …
- Mid-level
  - discrete pre/post-condition,
  - time pre/post-condition,
  - normalization, abstraction
  - garbage-collection (GC), protection from GC
- High-level
  - fwd/bck reachability,
  - TCTLF model-checking,
  - simulation-checking

Model construction
A while loop

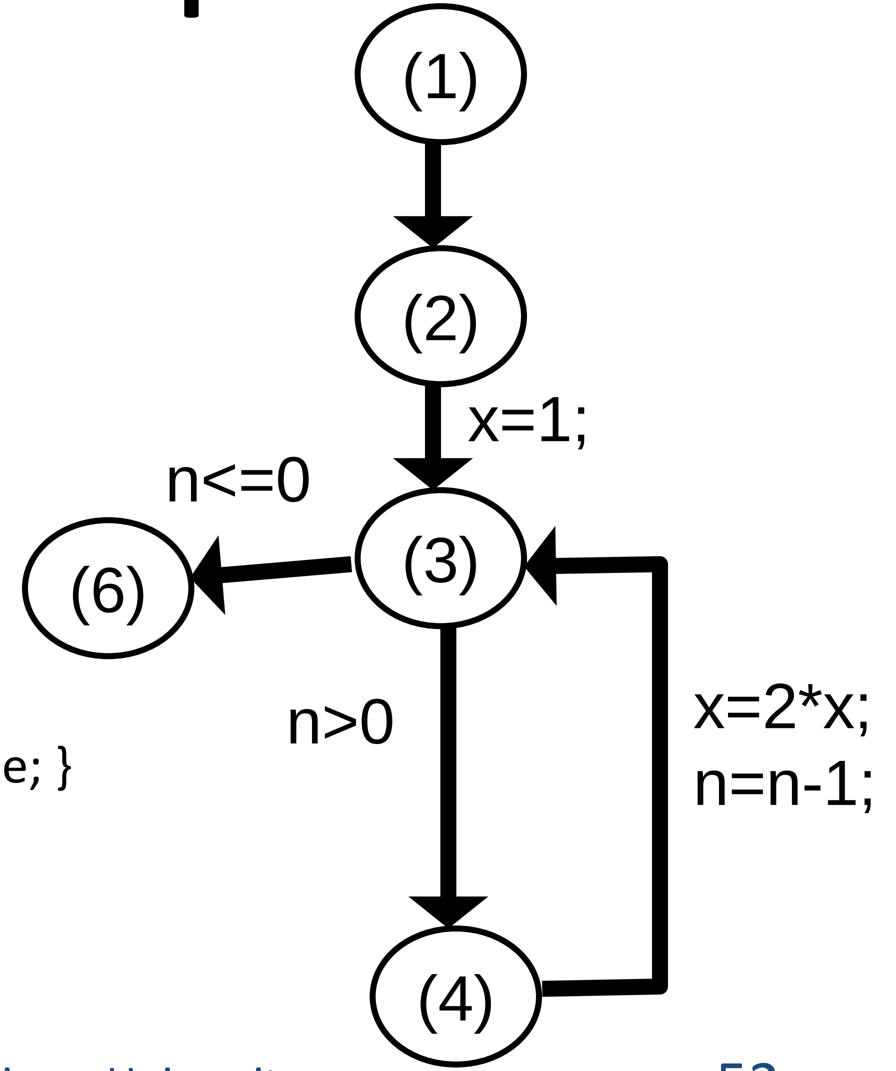
```
(1) f(n) {
(2)   x =1;
(3)   while n > 0, do {
(4)      x = x*2; n=n-1;
(5)   }
(6)   return x;
(7) }
```



Model construction
A while loop

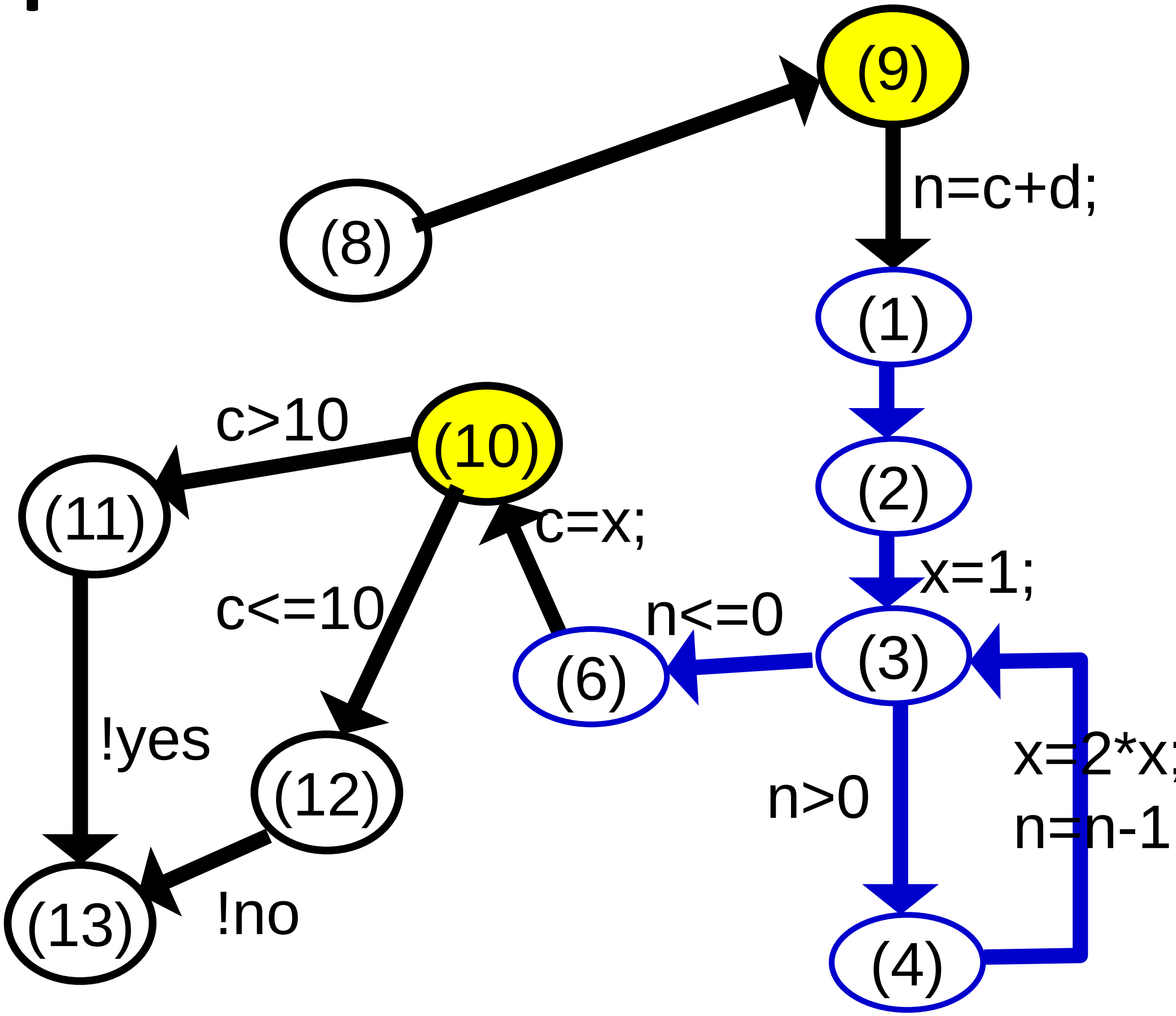
```
process count=1;
global discrete n:0..5;
global discrete x:0..32;
mode one (true) {
  when (true) may goto two; }
mode two (true) {
  when (true) may x=1; goto three; }
mode three (true) {
  when (n>0) may goto four;
  when (n<=0) may goto six; }
mode four (true) {
  when (true) may x=2*x; n--1; goto three; }
mode six (true) {  }

initially one@(1);
```



Model construction
A procedure call

```
(1)   f(n) {
(2)     x =1;
(3)     while n > 0, do {
(4)        x = x*2; n=n-1;
(5)     }
(6)     return x;
(7)   }
(8)   main (c,d) {
(9)     c = f(c+d);
(10)   if (c > 10)
(11)     print "yes".
(12)   else print "no".
(13) }
```



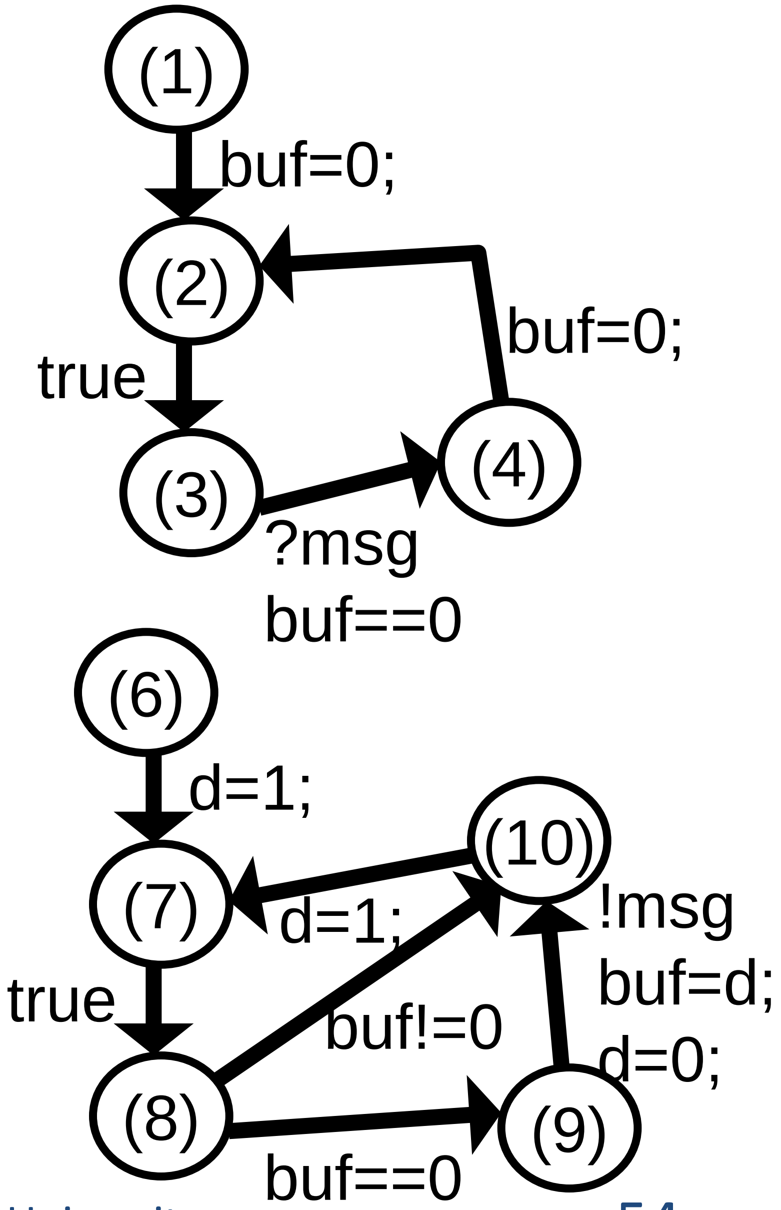
Model construction
Synchronization

The reader process

```
(1)  buf = 0;
(2)  while true, do {
(3)     if (buf == 0), read;
(4)     buf = 0;
(5)  }
```

The writer process

```
(6)  d = 1;
(7)  while true, do {
(8)     if (buf == 0),
(9)         write d to buf;d=0;
(10)    d = 1;
(11) }
```



CSMA/CD protocol (1/2)

```
#define  A          26
#define  B          52
#define  LAMBDA808
process count = 3; /* 1 is for bus, the others for senders. */
local clock x;
global synchronizer begin, end, cd, busy;
/* The following 3 modes are for the bus. */
mode idle (true) {
    when ?begin (true)  may x= 0; goto busy;
}
mode busy (true) {
    when ?end (true)  may x= 0; goto idle;
    when ?begin (x < A)  may x= 0; goto collision;
}
mode collision (x < A) {
    when !(#PS-1)cd (x < A)  may x= 0; goto idle;
}
```

constants

thread count

variables

oval,
mode,
location

arrow,
transition

CSMA/CD protocol (2/2)

```
/* The following 3 modes are for the senders. */
mode wait (true) {
    when !begin (true) may x= 0; goto send;
    when ?cd (true) may x= 0;
    when ?cd (true) may x= 0; goto retry;
}
mode send (x <= LAMBDA) {
    when !end (x==LAMBDA) may x= 0; goto wait;
    when ?cd (x<B) may x= 0; goto retry;
}
mode retry (x < B) {
    when !begin (x < B) may x= 0; goto send;
    when ?cd (x < B) may x= 0;
}
initially     idle@(1)&& x@(1) == 0
          &&  forall i:2..#PS, (wait@(i)&& x@(i) == 0);

risk  send@(2) && send@(3) && (x@(2) >= B||x@(3) >= B);
```

parameterized declaration

REDLIB application examples Sudoku (1/2)

```
#include <stdlib.h>
#include "redlib.h"
#include "redlib.e"
char *s[9][9];   int c[9][9];   int count_print = 0;
redgram  slot_constraint(d, i, j)  redgram d;   int i, j;  {
  int value, h, k, nc, ac, gs;  redgram conj;
  for(value =1; value <= 9; value++) /* To enumerate the value of s[i][j]. */ {
    for (h = 0; h < 9; h++){
      if (h != i) {
       // regulation for s[i][j] != s[h][j] in the same column.
       conj = red_diagram("!(%s==%d && %s==%d)", s[i][j], value, s[h][j], value);
        if ((++count_print) < 100 && (gs = red_diagram_size(d, &nc, &ac)) < 30) {
          fprintf(RED_OUT, "\nMutual exclusion to value:%1d at %s and %s:\nconstraint:\n",
            value, s[i][j], s[h][j]
          );
```

REDLIB application examples Sudoku (2/2)

```
     red_print_line(conj); fprintf(RED_OUT, "\n"); red_print_diagram(conj);
     fprintf(RED_OUT, "input diagram:\n");  red_print_line(d);
     fprintf(RED_OUT, "\n"); red_print_diagram(d);
    }
    d = red_and(d, conj);
    if ((h/3) == (i/3)) {
     for (k = 0; k < 9; k++) {
      if ((k/3)==(j/3) && (k!=j)) {
       d = red_and(d, red_diagram("!(%s==%d && %s==%d)", s[i][j], value, s[h][k], value) );
   } } } }
   if (h != j) {
    d = red_and(d, red_diagram("!(%s==%d && %s==%d)", s[i][j], value, s[i][h], value) );
 } } }
 h = red_push(d); red_garbage_collect(RED_GARBAGE_SILENT); red_pop(h); return(d);
}
 /* slot_constraint() */
```

REDLIB application examples red (1/4)

```
main(argc, argv)  int  argc; char **argv;  {
 redgram  sub, conj;   char *spec;
 if (!my_process_command_line(argc, argv) /* the number of files */)
  fprintf(RED_OUT, "Use a line beginning with \"%%end\" to end the formula input.\n\n”);
 red_begin_session(flag_my_system_type, model_fname, my_proc_count);
switch (task_type) {
 case SIM_CHECK: case BISIM_CHECK:
  red_input_model(model_fname, RED_NO_REFINED_GLOBAL_INVARIANCE);  break;
 default:
  if (flag_analysis_direction == FLAG_ANALYSIS_FORWARD)
   red_input_model(model_fname, RED_NO_REFINED_GLOBAL_INVARIANCE);
  else
   red_input_model(model_fname, RED_REFINE_GLOBAL_INVARIANCE);
 }
```

REDLIB application examples red (2/4)

```
  spec = red_file_to_string(spec_fname);
  my_verifier(task_type, spec);
  red_end_session(model_fname);
} /* main() */
```

REDLIB application examples red (3/4)

```
int  my_verifier(tt, s) int  tt/*task type */;  char *s/* string for the spec.*/ ;  {
  int  NEGATED_SPEC, assume, pi, xi, deadlock, wreach;
  struct reachable_return_type          *rr;
  struct sim_check_return_type          *sr;
  struct model_check_return_type        *mr;
  redgram                               result, ds;
```

REDLIB application examples red (4/4)

```
switch (tt) {
case SIM_CHECK:
 sr = red_sim_check(
  red_query_diagram_initial(), red_query_diagram_global_invariance(),
  RED_FULL_REACHABILITY, RED_NO_REACHABILITY_DEPTH_BOUND,
  flag_counter_example, RED_TIME_PROGRESS,
  flag_normality, flag_action_approx,
  flag_reduction, flag_approx, flag_symmetry, flag_zeno,
  flag_tconvexity_shared_partitions  | flag_time_progress_options
  | flag_gfp_on_the_fly | flag_fairness_assumptions_eval
  | flag_gfp_path,
  flag_print, s
 );
 red_print_sim_check_return(sr); return (sr->iteration_count);
 break;
```

REDLIB application examples some API (1/2)

A Procedure for calculating the precondition of synchronous transitions with fine-resolution control.

```
redgram red_sync_xtion_bck(
  redgram                ddst,
  redgram                dpath,
  int                    flag_sync_xtion_table_choice,
  int                    sxi,
  int                    flag_game_roles,
  int                    flag_time_progress,
  int                    flag_normality,
  int                    flag_action_approx,
  int                    flag_reduction,
  int                    flag_state_approx,
  int                    flag_symmetry,
  int                    flag_experiment
);
```

REDLIB application examples some API (2/2)

```
redgram red_norm(
  redgram        d,
  int            op
);
```

A procedure for normalizing decision diagrams.

op: option for normal forms.

```
redgram red_abstract(
  redgram        d,
  int            flag_oapprox,
  char           *role_spec,
  …
);
```

A procedure for abstracting decision diagrams.

op: abstraction resolution based on roles.

three roles: model, spec, envr.

Potential research issues

- PAT input → REDLIB input
  - What is the best translation for verification perfomrance ? Broadcasting semantics ?
  - Pre-analysis of event models ?
- REDLIB output → PAT output
- Garbage-collection for CRD+MDD ?
- Probabilistic model-checking with REDLIB
  - how to outperform prism (DBM+BDD) ?
- Bound analysis ?