Difference Decision Diagrams

Jesper Møller, Jakob Lichtenberg,
Henrik Reif Andersen & Henrik Hulgaard

The IT University of Copenhagen
Glentevej 67 • 2400 Copenhagen • Denmark

The Logic

A first-order logic over difference inequalities:

$$\phi \ ::= \ x - y \leq d \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \exists x.\phi$$

We want to solve the decision problem:

SAT: is ϕ satisfiable?

Notice:

- No absolute constraints. Can be introduced using a variable z representing "zero".
- No Boolean variables. Can be modelled by $x - x' \leq 0$.

Using the Logic

IT-højskolen i København

The logic naturally induces a guarded command language:

$$\phi \rightarrow x := \mathbf{any}\ x'.\ \phi$$

(With obvious abbreviations: $x := z + c$ is $x := \mathbf{any}\ x'.\ x' = z + c$.)

This *timed guarded command language* can be used for modelling dynamic systems with continuous time.

Some of the variables are *clocks*, others model discrete state (e.g. locations) through a Boolean encoding.

A decision procedure for the logic makes it possible to verify such systems.

Symbolic State-Space Exploration

IT-højskolen i København

Represent a set of states as a formula ϕ.

$\phi \leftarrow \phi_0$
$R \leftarrow \phi$
while $\phi \not\equiv \mathit{false}$ **do**
  $\phi \leftarrow \mathsf{Next}(\phi) \wedge \neg R$
  $R \leftarrow R \vee \phi$

$$R \;=\; \Big((s_0 \vee s_1) \wedge 0 \leq x - y < 5\Big) \vee \cdots$$

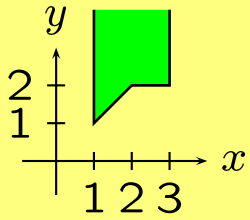
We need : 1. A logic to express formulas in.
2. Next(ϕ).
3. A data structure for representing formulas.

Overview

IT-højskolen i København

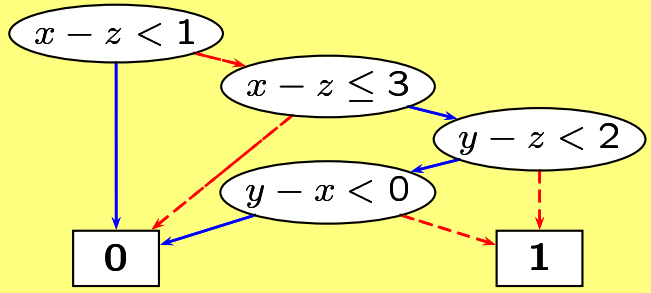
Difference Constraint Expressions

$\phi ::= x - y \leq c \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \exists x.\phi$



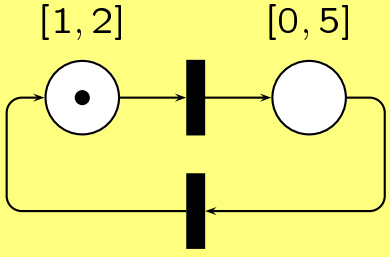
Representation

Difference Decision Diagrams

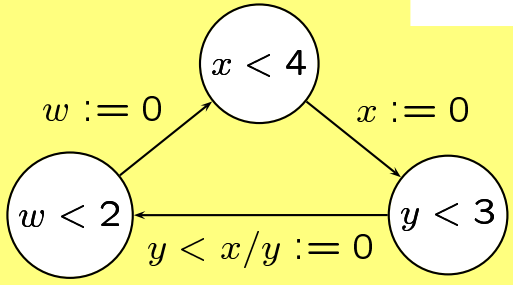


Verification

Timed Petri Nets



Timed Automata



Timed Guarded Commands

$H - z \geq 25 \;\rightarrow\; z := \textbf{any}\ z'.$
$h_i \Rightarrow H - z' \leq 200$

Difference Decision Diagrams

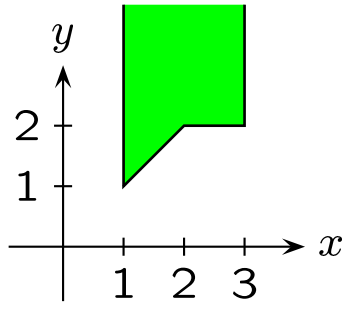
IT-højskolen i København

Implicit representation of the logic:

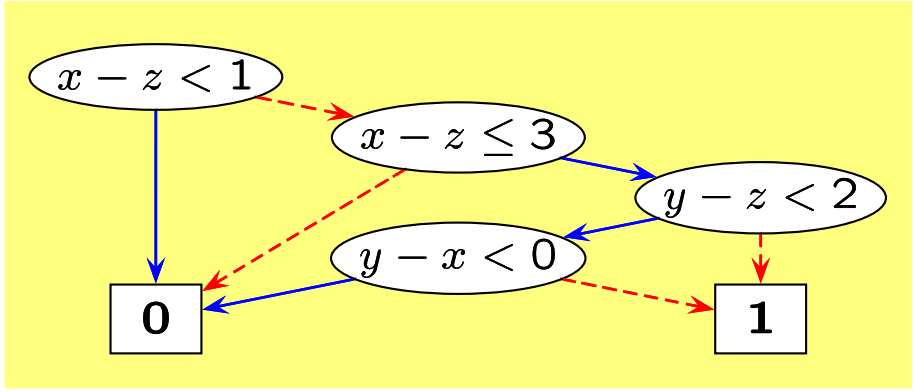
$$\phi \ ::= \ x - y \leq d \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \exists x.\phi\,.$$

Example

$$\phi \ = \ 1 \leq x - z \leq 3 \ \wedge \ (y - z \geq 2 \ \vee \ y - x \geq 0)$$



(x, y)-plot for $z = 0$

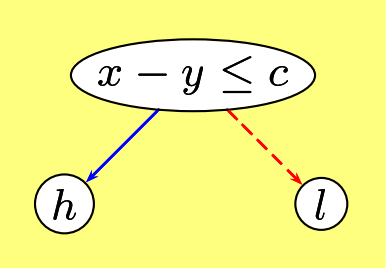


Difference decision diagram

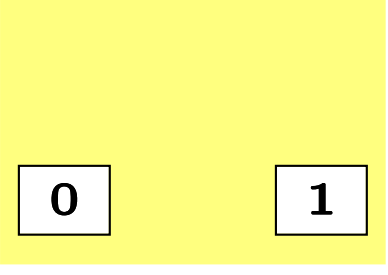
Local Reduction Rules

IT-højskolen i København

A *Difference Decision Diagram* is a directed, acyclic graph with two types of nodes:



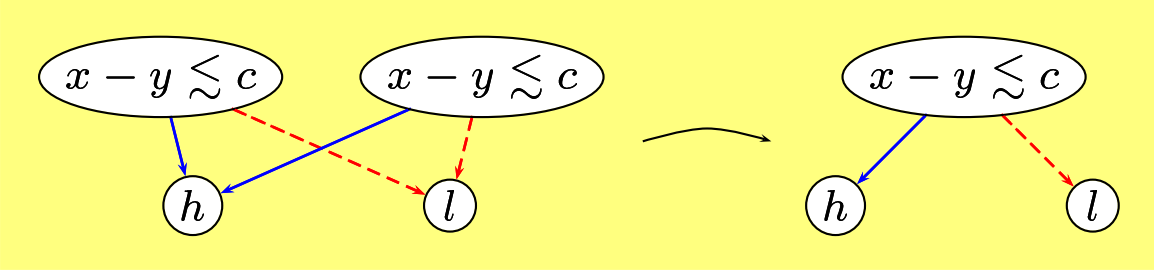
if $x - y \leq c$ then h else l



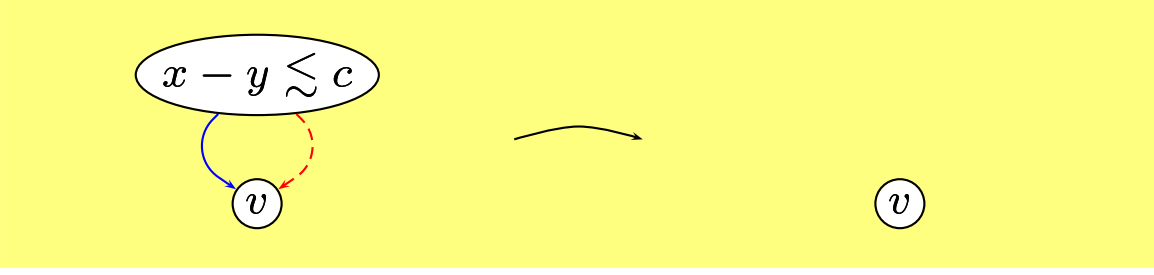
false respectively true

Local Reduction Rules

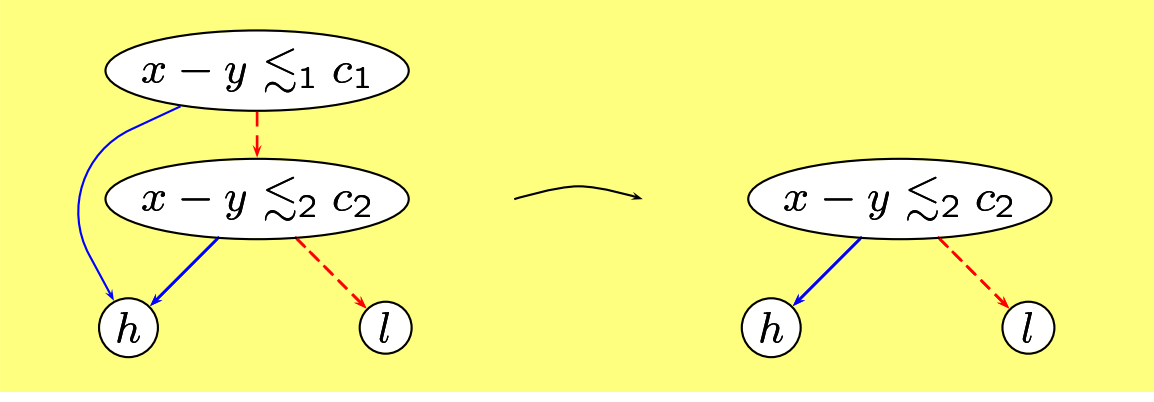
IT-højskolen
i København



No duplicate vertices

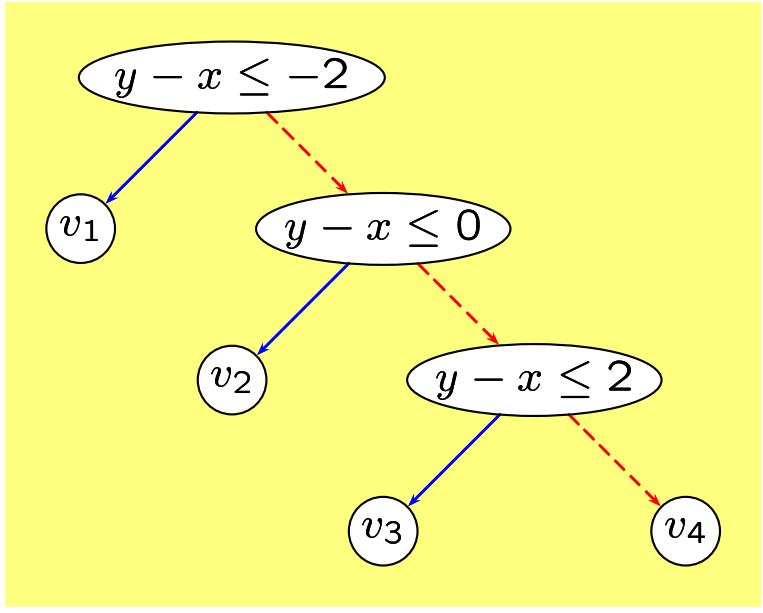


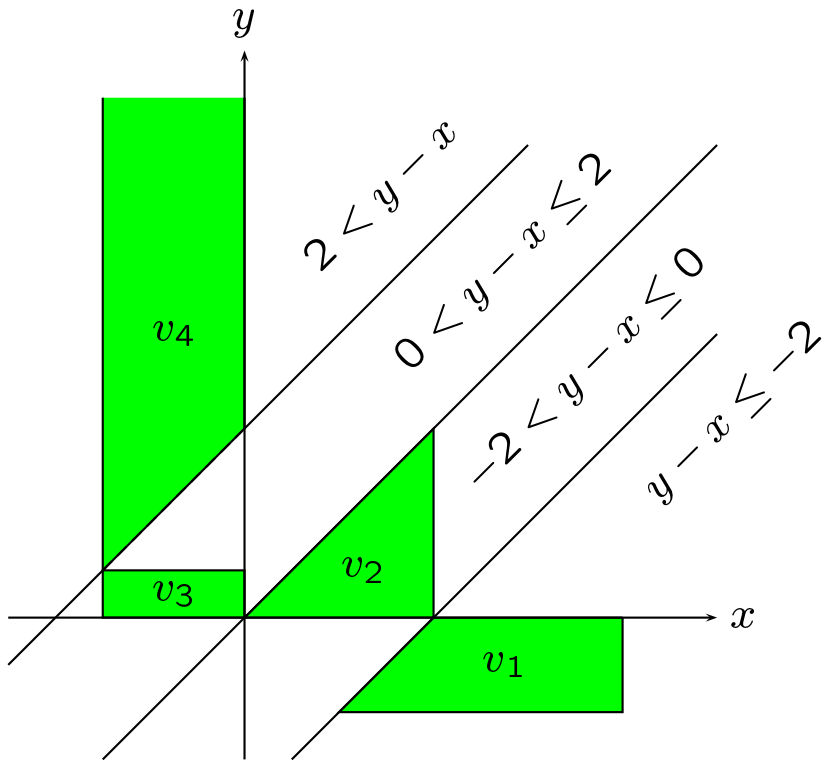
Different high- and low-branches



No redundant tests

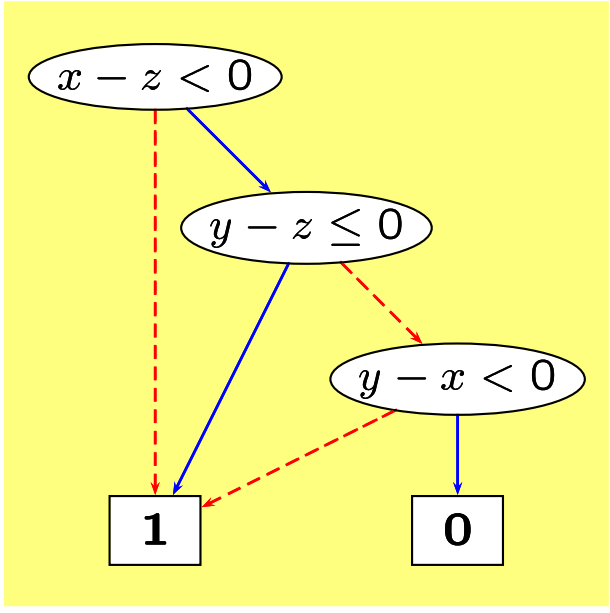
An Ordered DDD



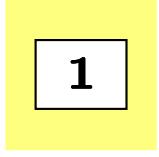


Path Reduction

IT-højskolen
i København



Locally Reduced



Path Reduced

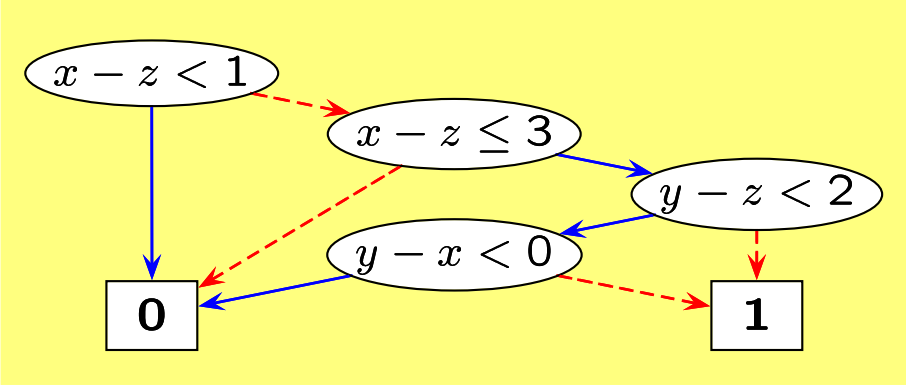
Semi-Canonicity

IT-højskolen i København

A *path-reduced* DDD is a locally reduced DDD where all paths are feasible.

Theorem

A path-reduced DDD is unsatisfiable if and only if it is the terminal vertex $\mathbf{0}$.

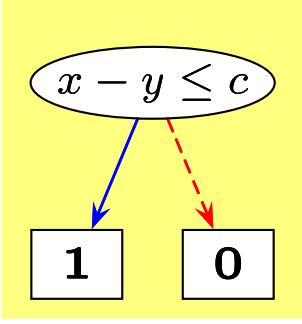


Constructing and Manipulating DDDs

IT-højskolen i København

Constructing:

$x - y \leq c$	$\phi_1 \wedge \phi_2$	$\neg\phi$	$\exists x.\phi$
$mk(x, y, c, 1, 0)$	$apply(\wedge, u_1, u_2)$	$apply(\neg_1, u, \_)$	$exists(x, u)$



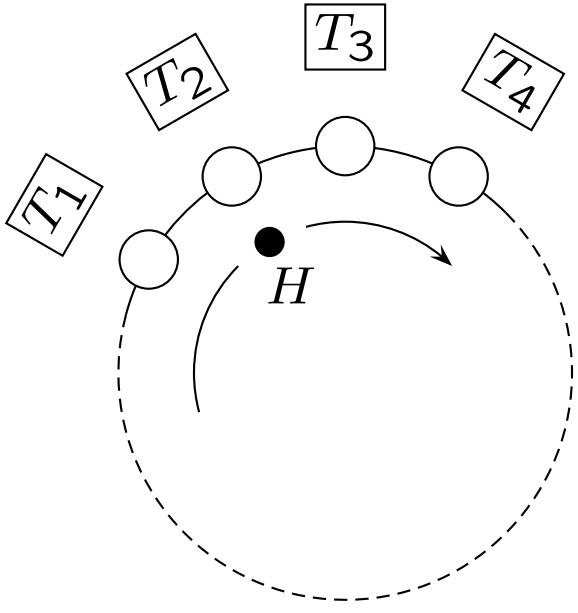
$apply(op, u_1, u_2)$ "merges" two DDDs into one utilizing the ordering

Fourier-Motzkin quantifier elimination

SAT-testing:

Path reduce, check whether it is the terminal $\mathbf{0}$.

Milner's Scheduler [Milner 89]



$$
\begin{array}{lcll}
c_i \wedge \neg t_i & \rightarrow & H, T_i, t_i, c_i, h_i & := 0, 0, \mathsf{T}, \mathsf{F}, \mathsf{T} \\
h_i \wedge H^l \leq H & \rightarrow & c_{(i \bmod N)+1}, h_i & := \mathsf{T}, \mathsf{F} \\
t_i \wedge T^l \leq T_i & \rightarrow & t_i & := \mathsf{F}
\end{array}
$$

$$I = \bigwedge_{i=1}^{N} (h_i \Rightarrow H \leq H^u) \wedge (t_i \Rightarrow T_i \leq T^u)$$

Experimental Results

IT-højskolen i København

N	Kronos	Uppaal	DDD
4	0.2	0.1	0.1
5	0.7	0.2	0.1
6	22.6	0.6	0.1
7	339.2	2.3	0.1
8	–	9.0	0.2
9	–	35.0	0.2
10	–	138.4	0.2
11	–	529.8	0.2
12	–	2560.7	0.3
16	–	–	0.5
32	–	–	2.2
64	–	–	15.9
128	–	–	123.3
256	–	–	1104.8

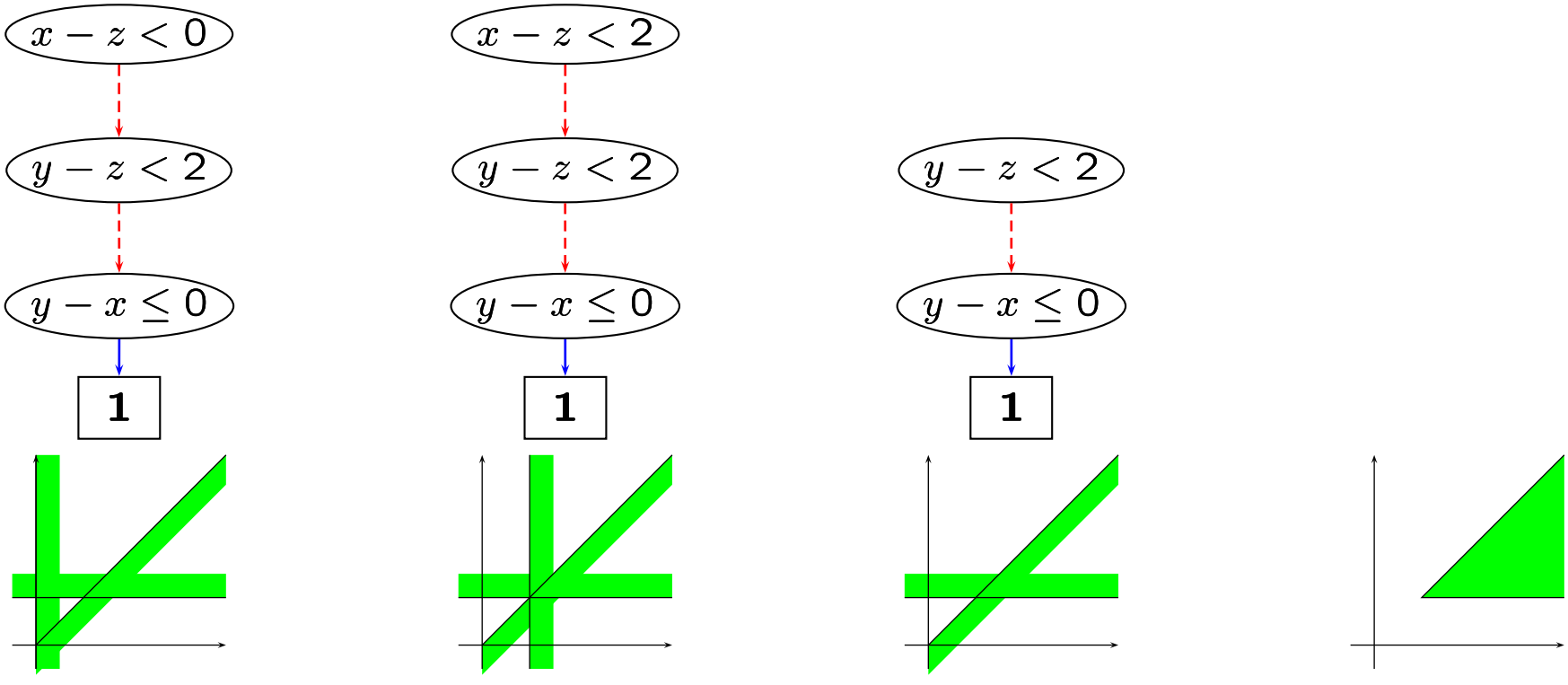
(a) 1 clock.

N	Kronos	Uppaal	DDD
4	0.4	0.2	0.2
5	2.4	1.7	0.3
6	24.2	17.6	0.5
7	346.6	201.7	0.5
8	–	2460.2	0.6
16	–	–	1.5
32	–	–	5.7
64	–	–	31.7
128	–	–	217.3

(b) $N + 1$ clocks.

Fully Reduced, Canonical DDDs?

IT-højskolen
i København



Conclusion

IT-højskolen
i København

- Symbolic model checking of timed systems requires a data structure and a decision procedure for the logic :

$$\phi \ ::= \ x - y \leq d \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \exists x.\phi \, .$$

- Advancing time amounts to existential quantification of the z-variable.

- TCTL formulas can be verified by letting time go backwards.