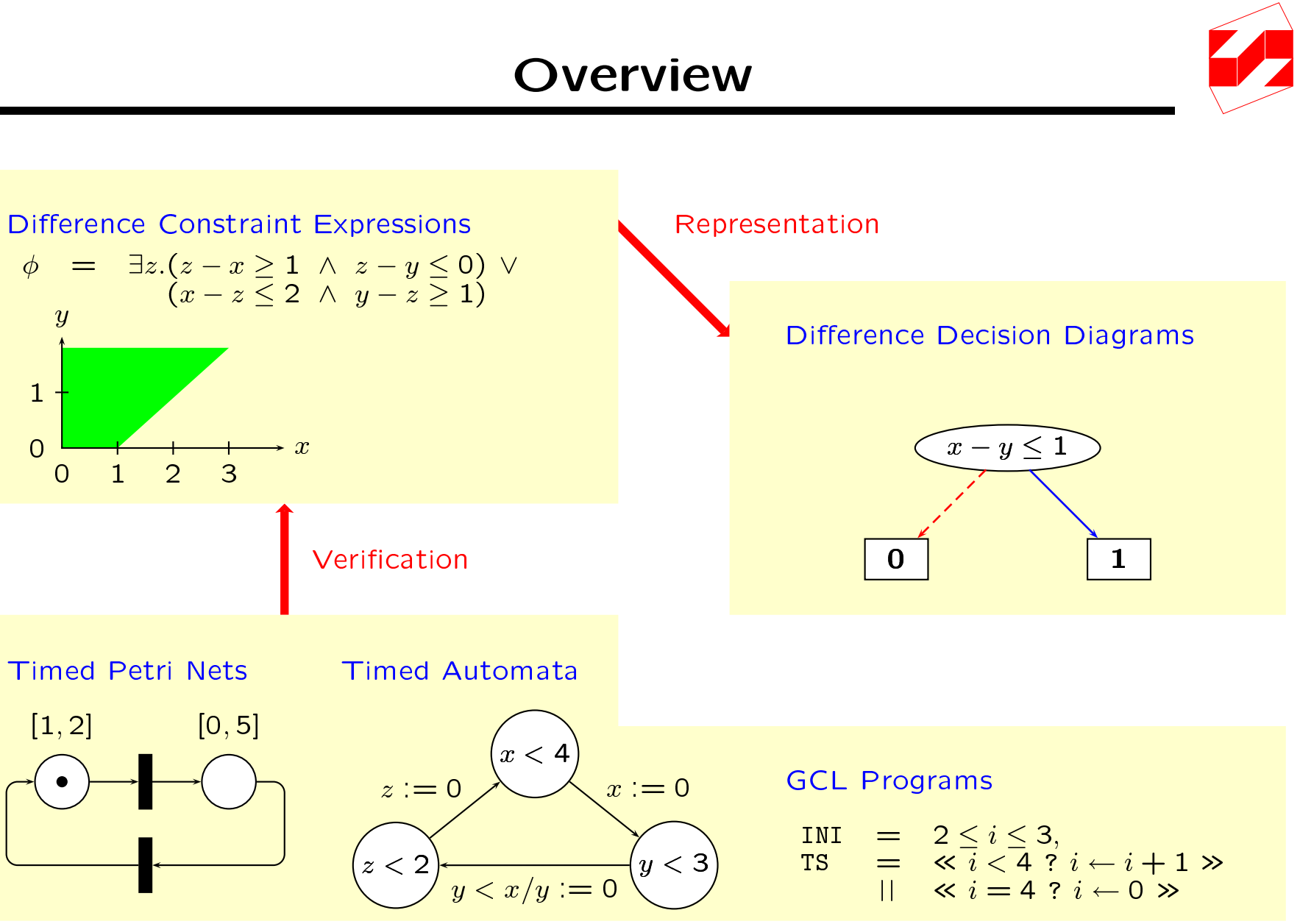
Difference Decision Diagrams

Jesper Møller and Jakob Lichtenberg
Department of Information Technology
Technical University of Denmark
Building 343, DK-2800 Lyngby
{jmr,jali}@it.dtu.dk

October 5, 1998

Overview


Difference Constraint Expressions
$\phi = \exists z.(z - x \geq 1 \wedge z - y \leq 0) \vee (x - z \leq 2 \wedge y - z \geq 1)$
y
1
0
x
0 1 2 3
Representation
Difference Decision Diagrams
$x - y \leq 1$
0
1
Verification
Timed Petri Nets
[1, 2]
[0, 5]
Timed Automata
$x < 4$
$z := 0$
$x := 0$
$z < 2$
$y < 3$
$y < x/y := 0$
GCL Programs
INI = $2 \leq i \leq 3$,
TS = « $i < 4$? $i \leftarrow i + 1$ »
|| « $i = 4$? $i \leftarrow 0$ »

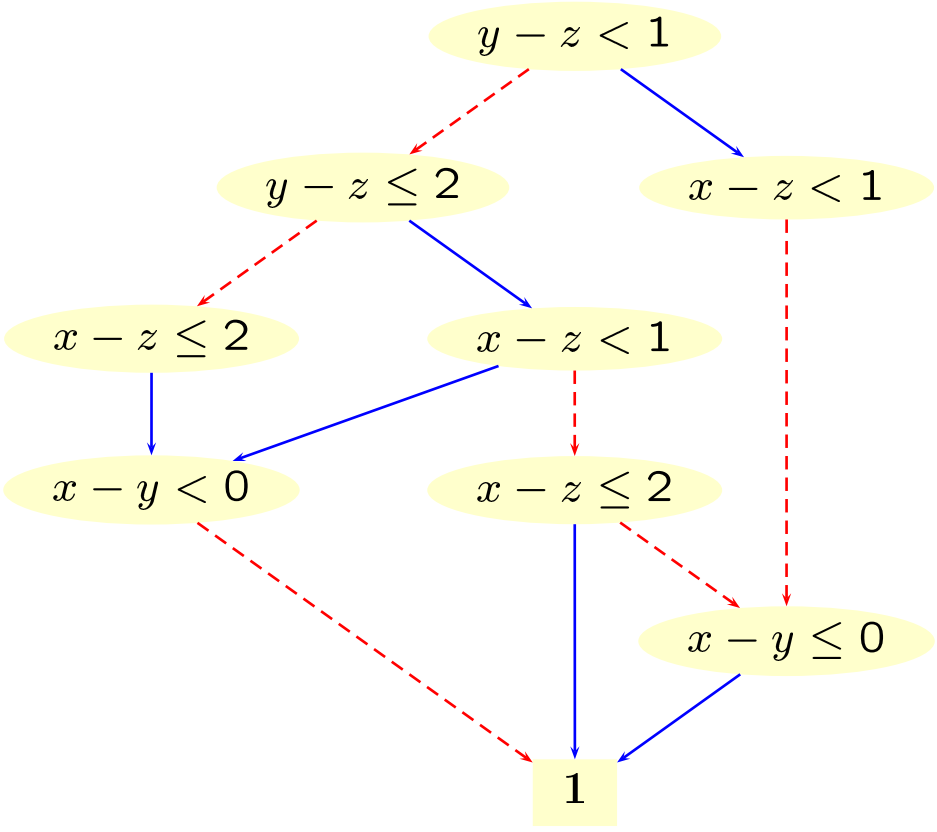
Definition of a DDD

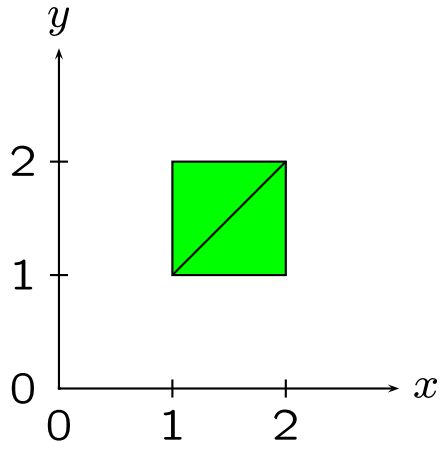
Definition 1 A difference decision diagram (DDD) is a directed, acyclic graph with:

- Two terminal vertices: **0** and **1**.
- A set of non-terminal vertices:

$$x_i - x_j < c \longrightarrow h, l \quad \text{or}$$
$$x_i - x_j \leq c \longrightarrow h, l$$

An Example





Definition of an Ordered DDD

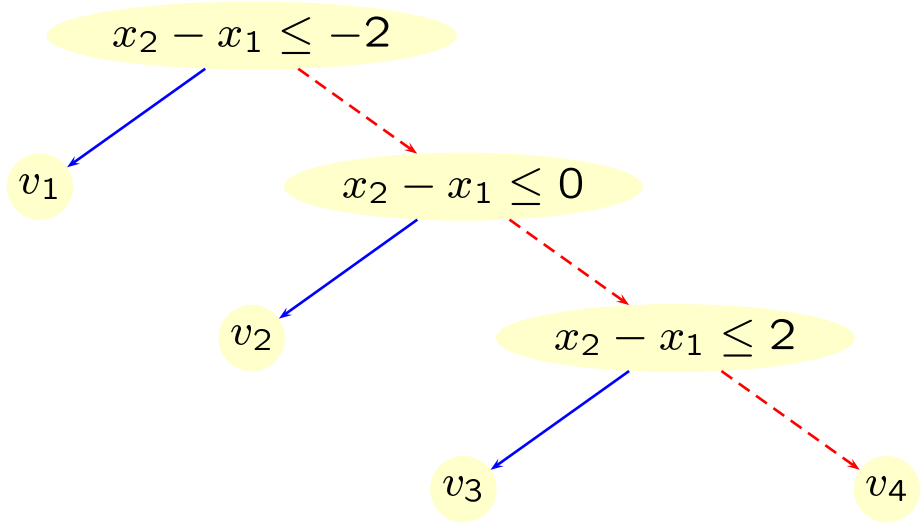
Definition 2 An ordered DDD v satisfies:

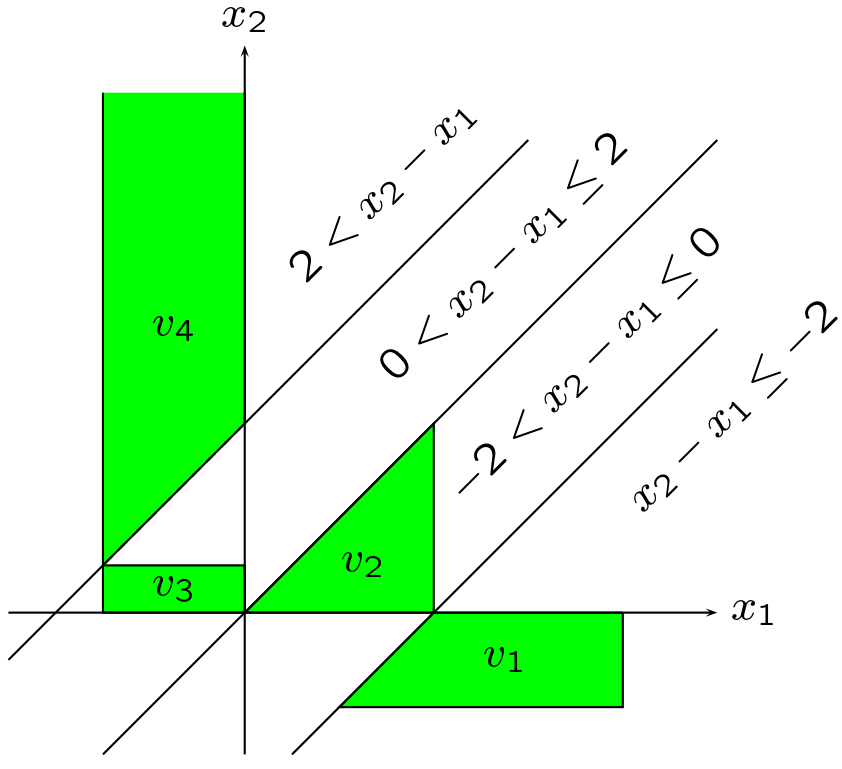
1. $neg(v) < pos(v)$,

2. $var(v) < var(high(v))$,

3. $var(v) < var(low(v)) \ \vee$
   $var(v) = var(low(v)) \ \wedge \ bound(v) < bound(low(v))$.

An Ordered DDD





Definition of a Locally Reduced DDD

Definition 3 A locally reduced DDD satisfies for all non-terminals u and v:

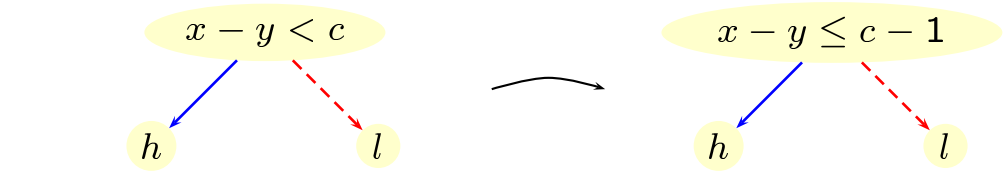
1. $\mathbb{D} = \mathbb{Z}$ implies $op(v) = \leq$,

2. $attr(u) = attr(v)$ implies $u = v$,

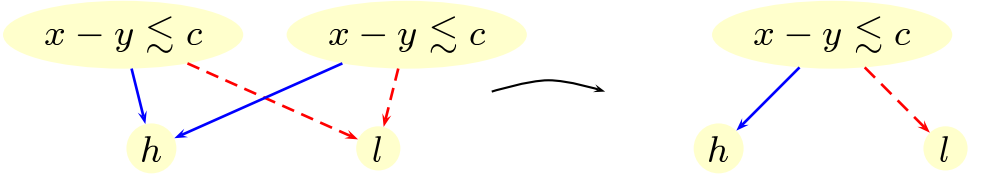
3. $low(v) \neq high(v)$,

4. $var(v) = var(low(v))$ implies $high(v) \neq high(low(v))$.

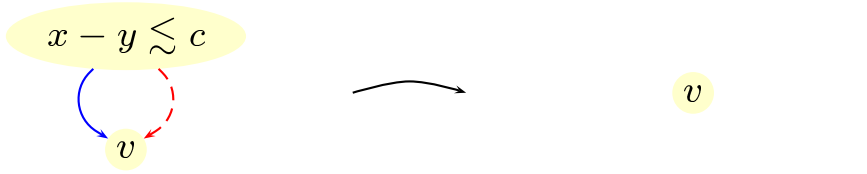
Local Reduction Requirements



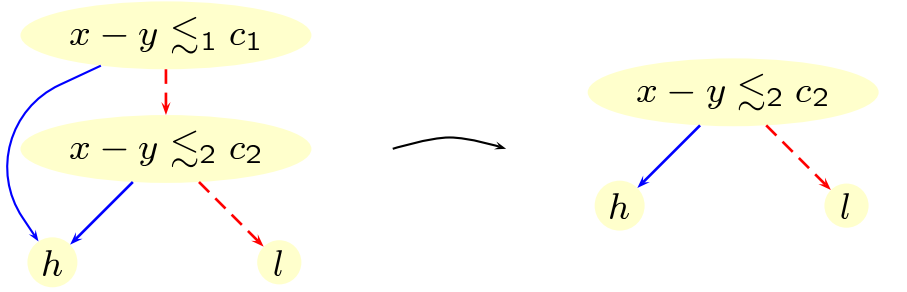
If $\mathbb{D} = \mathbb{Z}$, strict constraints must be expressed as weak constraints.



Duplicate vertices must not be present.



Vertices with the same high- and low-branches must not be present.



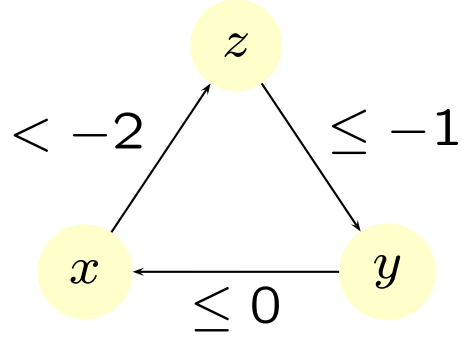
Redundant test must not be present.

Definition of a Path Reduced DDD

Definition 4 A path reduced DDD contains no infeasible paths.

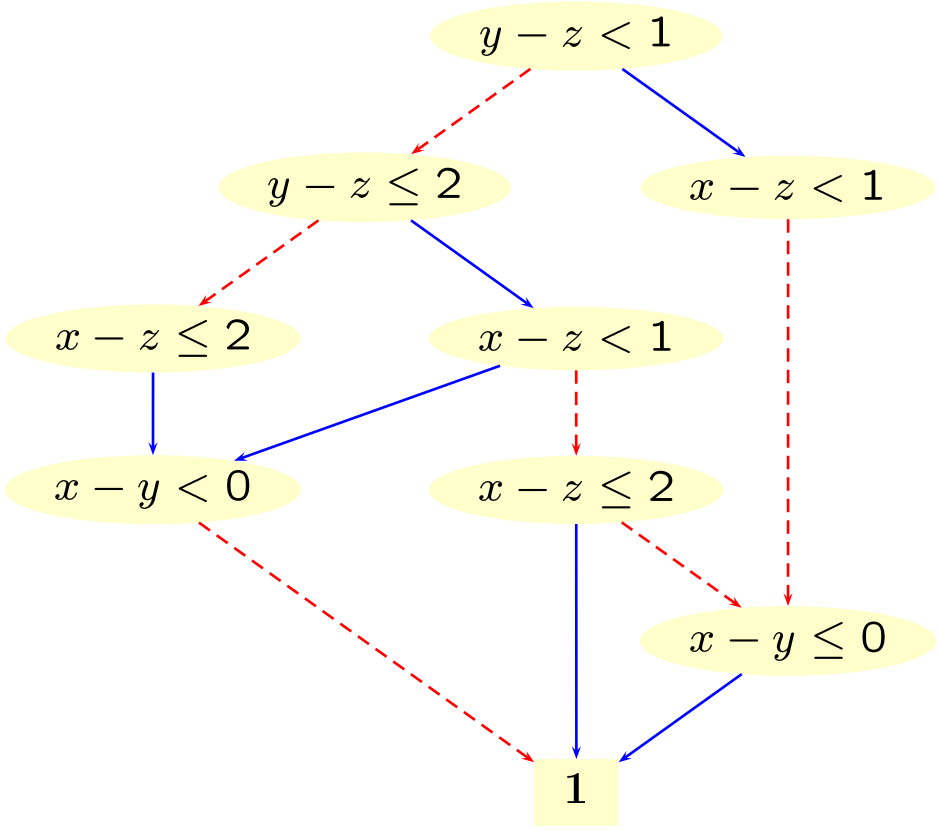
A path defines a constraint system and a constraint graph, for example:

$$\begin{array}{rcl} x - y & \leq & 0 \\ z - x & < & -2 \\ y - z & \leq & -1 \end{array}$$

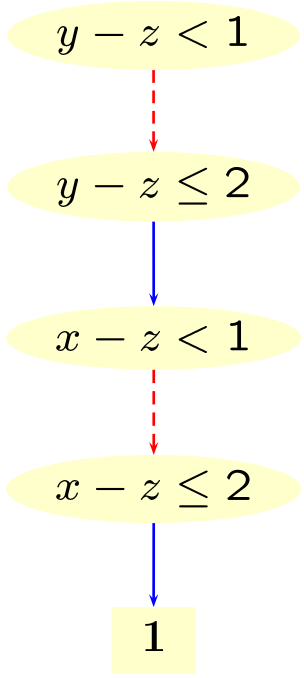


Lemma 2 A constraint system is feasible if and only if its constraint graph has no negative-weight cycles.

An Example of Path Reduction



Locally reduced



Path reduced

Levels of Canonicity

Locally reduced DDDs are not canonical.

Theorem 11 Path reduced DDDs are semi-canonical:

$$\begin{array}{ll} [\![u]\!] = \texttt{true} & \text{if and only if } \; u = \mathbf{1} \\ [\![u]\!] = \texttt{false} & \text{if and only if } \; u = \mathbf{0} \\ [\![u]\!] \neq \texttt{true} & \text{if and only if } \; u \neq \mathbf{1} \\ [\![u]\!] \neq \texttt{false} & \text{if and only if } \; u \neq \mathbf{0} \end{array}$$

Conjecture 14 Fully reduced DDDs are canonical:

$$[\![u]\!] = [\![v]\!] \;\; \text{if and only if } \;\; u = v$$

Relationship between DCEs and DDDs

Difference Constraint Expressions

$$
\begin{array}{rcl}
\phi & ::= & \texttt{false} \mid \texttt{true} \\
& \mid & x - y < c \mid x - y \leq c \\
& \mid & \neg\phi \\
& \mid & \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \rightarrow \phi_2 \mid \phi_1 \leftrightarrow \phi_2 \\
& \mid & \exists x.\phi \mid \forall x.\phi
\end{array}
$$

DDD operations

$\mathbf{0}, \mathbf{1}$	
MkDiffCstr	
Not	
Apply	
Exists, Forall	

Functional Properties

Tautology	$(\models \phi)$	PathReduce(u_ϕ)	$=$	$\mathbf{1}$
Satisfiability		PathReduce(u_ϕ)	$\neq$	$\mathbf{0}$
Equivalence	$(\phi \Leftrightarrow \omega)$	PathReduce(Apply$(\leftrightarrow, u_\phi, u_\omega)$)	$=$	$\mathbf{1}$

Efficiency

Difference constraint expressions:

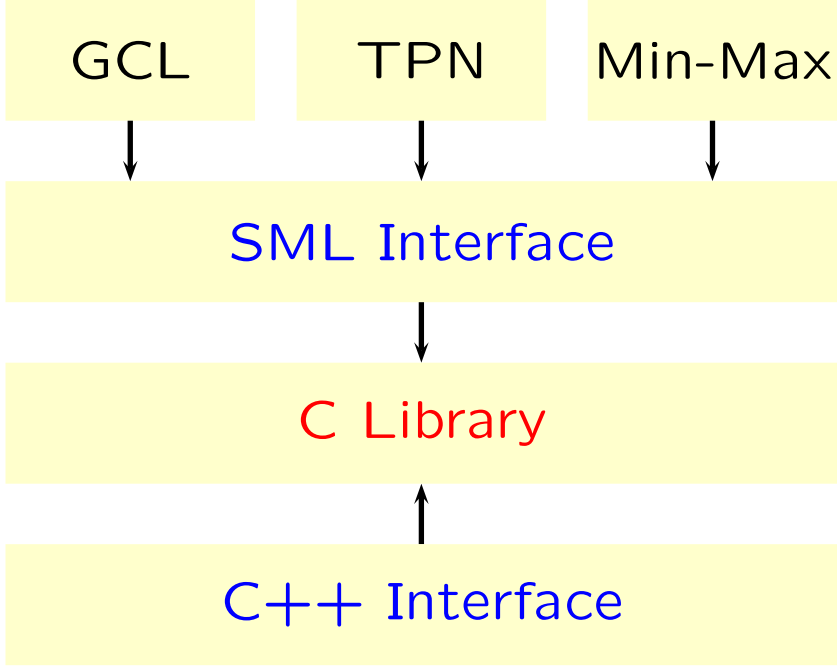
Theorem 5 Satisfiability of a quantifier-free difference constraint expression is NP-complete.

Lemma 6 Satisfiability of a quantified difference constraint expression is PSPACE-hard.

Difference decision diagrams:

Algorithm	*Complexity*
MkDiffCstr	Constant time
Not, Apply	Polynomial
Exists, Forall	Exponential
PathReduce	Exponential
Tautology, Satisfiability, Equivalence	Exponential

The DDD Tool



Static Program Analysis

McCarthy's 91 function $f : \mathbb{Z}_+ \to \mathbb{Z}_+$:

$$f(x) = \begin{cases} x - 10 & \text{if } x \geq 101, \\ f(f(x+11)) & \text{if } x < 101. \end{cases}$$

Verify that: $f(x) = 91$, for all $x = 0, 1, \ldots, 101$.

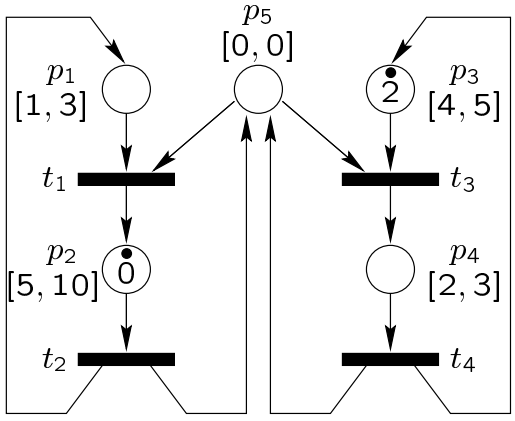
A GCL program for f:

$$\begin{array}{lcl} \texttt{INI} & = & (i = 1) \wedge \Big((n = 0) \vee \cdots \vee (n = 101) \Big), \\ \texttt{TS} & = & \ll (i \geq 1) \wedge (n \geq 101) \ \texttt{?} \ i \leftarrow i - 1; \quad n \leftarrow n - 10 \gg \\ & \texttt{||} & \ll (i \geq 1) \wedge (n < 101) \ \texttt{?} \ i \leftarrow i + 1; \quad n \leftarrow n + 11 \gg \end{array}$$

Verify the invariant: $(i = 0) \to (n = 91)$

Systems with Time—The Problem

Consider a timed Petri net:



$$(\{p_2, p_3\}, [x_2 \mapsto 0, x_3 \mapsto 2])$$

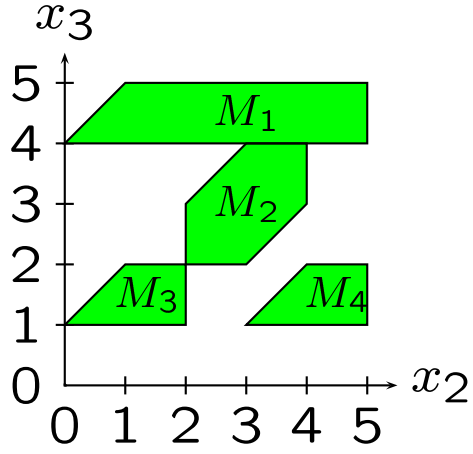
A state of a timed Petri net is a pair (μ, t), where μ is a marking and t is the timer values.

How can we represent sets of states?

How can we compute the set of reachable states?

The Standard Solution

A marking is represented as a bit-vector. Timer values are represented as a set of difference bound matrices (Rokicki 1993).



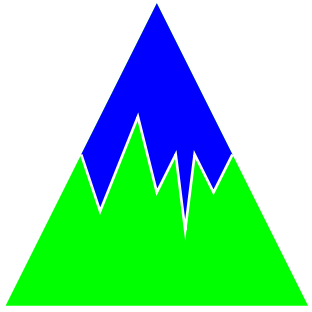
$(\texttt{01100}, \{M_1, M_2, M_3, M_4\})$

Disadvantages:

- Explicit representation
- Control and timing information are separated
- No sharing
- Matrix-specific operations

The Solution using DDDs

A set of states is represented as a DDD.



Markings

Timing information

Advantages:

- Implicit representation
- A single data structure
- One DDD for the entire state space
- High level of abstraction using difference constraint expressions

Conclusion

Contributions:

- A first-order logic for difference constraint expressions
- A specification of difference decision diagrams
- A number of applications of difference decision diagrams
- An implementation of difference decision diagrams

Further Research

- Improve theory
- Improve algorithms
- Combine DDDs with BDDs into a hybrid data structure
- Focus on applications