# Binary Constraint Trees and Structured Decomposability

Petr Kučera

Charles University, Czech Republic

Boolean Seminar 2023
Liblice, September 24–28, 2023

# Overview

# CSP Notation

**CSP** a pair $(\mathbf{x}, C)$
- variables $\mathbf{x}$
- constraints $C$

**Domain of $x \in \mathbf{x}$** finite set of values $dom(x)$

**Literal on $x \in \mathbf{x}$** assignment $(x, a)$, $a \in dom(x)$

**Tuple over $\{x_{i_1}, \ldots, x_{i_r}\}$** set of literals $\{(x_{i_1}, a_1), \ldots, (x_{i_r}, a_r)\}$

**Scope of $c \in C$** variables on which $c$ is defined

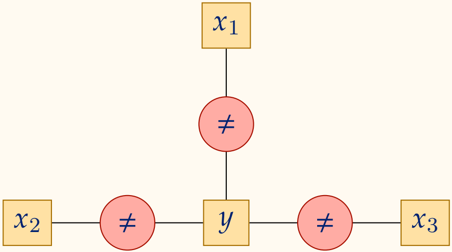
**Relation of $c \in C$** tuples that satisfy $c$

**Solution of $P$** tuple $\tau$ over $\mathbf{x}$ satisfying all constraints $c \in C$
- restriction of $\tau$ to the scope of $c$ belongs to the relation of $c$

# Binary Constraint Tree (Wang and Yap, 2022b)

Normalized binary CSP $(\mathbf{x}, C)$ whose constraint graph is a tree

Domains of all variables are $\{0, 1, 2\}$



# BCT Constraint (Wang and Yap, 2022b)

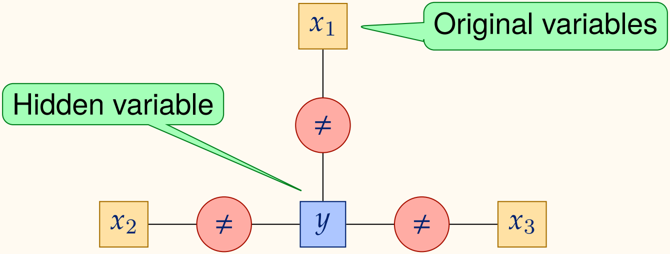
Pair $(\mathbf{x}, P)$ which consists of

| | |
|---|---|
| Binary constraint tree | $P = (\mathbf{z}, C)$ |
| Original variables | $\mathbf{x}$ |
| Hidden variables | $\mathbf{z} \setminus \mathbf{x}$ |
| Constraint relation | solutions of $P$ restricted to the original variables $\mathbf{x}$ |

# Not All Different

BCT constraint $x_1 = x_2 \lor x_1 = x_3 \lor x_2 = x_3$

Domains of all variables are $\{0, 1, 2\}$



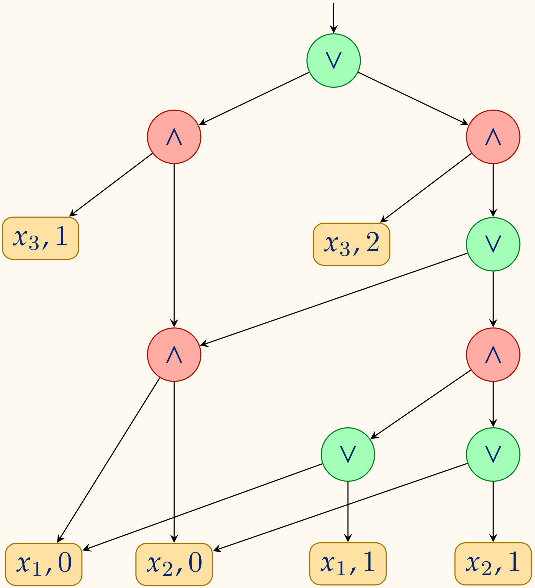
# Properties of BCT Constraints

- Efficient consistency checking an propagation
- MDD can be encoded as a BCT (Wang and Yap, 2022b)
- NFA constraint can be encoded as a BCT (Wang and Yap, 2022a)
- Propagation complete encodings of BCT constraints (Wang and Yap, 2022a)
- Efficient combinations of BCT constraints having the same tree structure (Wang and Yap, 2023)

**Our result**

BCT constraints are polynomially equivalent to constraints that can be represented with structured DNNFs.

# Multivalued Negation Normal Form (NNF)




Directed acyclic graph representing a circuit

Inner nodes labeled with $\wedge$ and $\vee$ gates

Leaves labeled with literals

# NNF Constraint

Constraint $c$ represented with a NNF $D$:

**Scope** variables in the leaves $D$

**Relation** tuples $\tau$ on which $D$ evaluates to true

- Set inputs $(x, a) \in \tau$ to true
- Set inputs $(x, a) \notin \tau$ to false

**Decomposable NNF (DNNF)** decomposable $\wedge$ gates

- Darwiche, 1999
- Efficient consistency checking
- Efficient propagation (Gange and Stuckey, 2012)

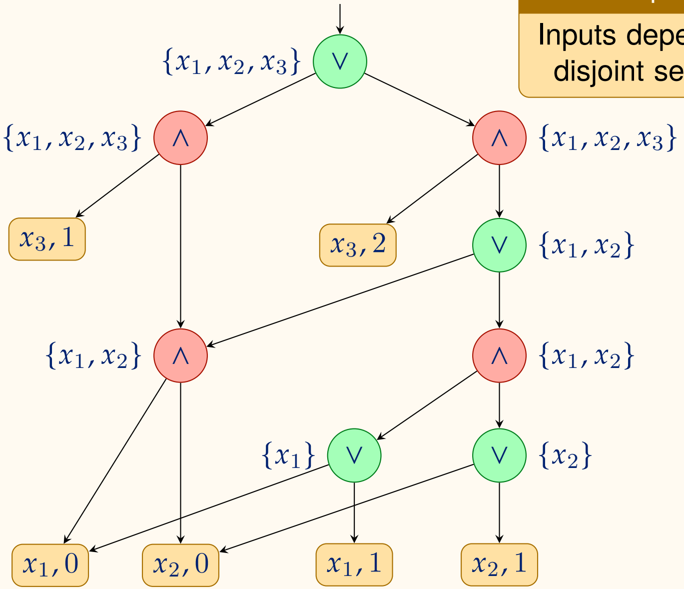
**Structured DNNF (SDNNF)** conjunctions have a tree-like structure

- Pipatsrisawat and Darwiche, 2008
- Efficient conjoining two SDNNFs with the same structure

# Decomposable NNF (DNNF)

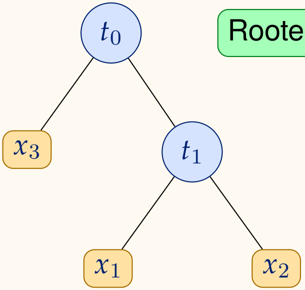
**Decomposable ∧ gates**

Inputs depend on pairwise disjoint sets of variables



# V-Tree

v-tree

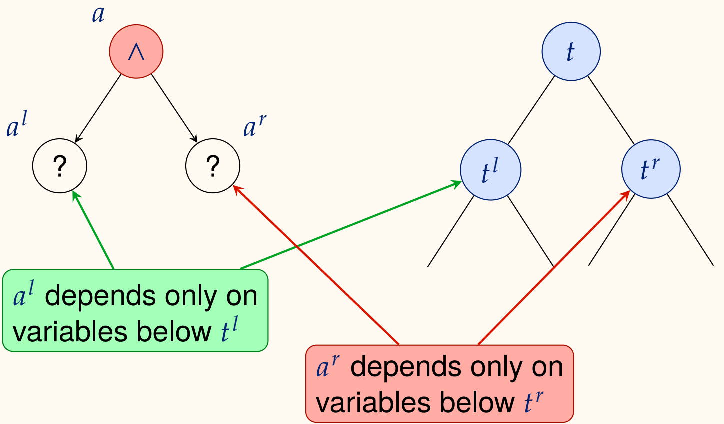



Rooted binary tree

Leaves identified with variables

# Conjunction Respecting a V-Tree Node

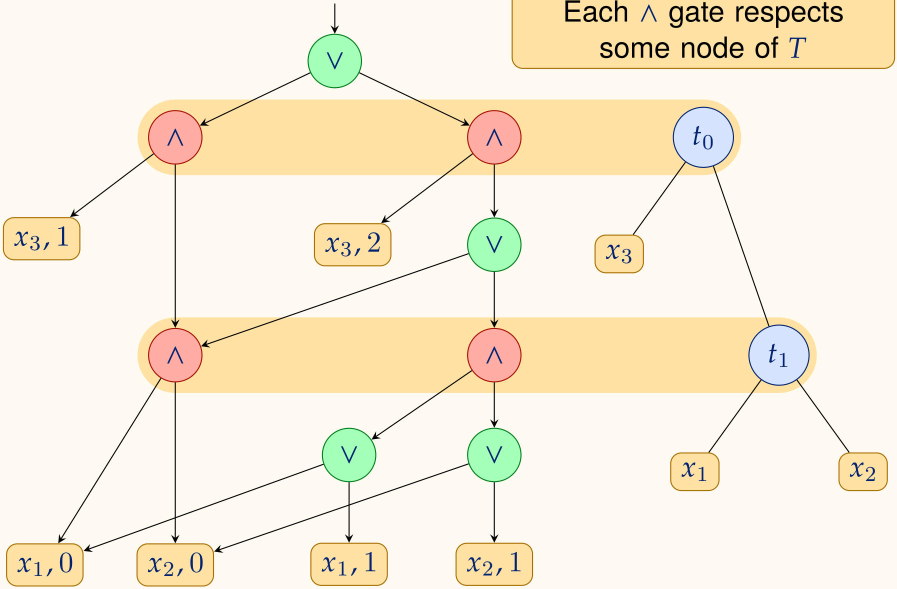
Conjunction gate $a$ respects $t$



# DNNF Respecting a V-Tree

**DNNF respects v-tree $T$**

Each $\land$ gate respects some node of $T$



# Structured DNNF (SDNNF)

**Pipatsrisawat and Darwiche, 2008**

DNNF $D$ is structured (SDNNF) if it respects some v-tree.

- Includes structured decision diagrams (OBDD, MDD, SDD)
- Strictly less succinct than DNNF
- Strictly more succinct than AOMDD
- Efficient conjoining two SDNNF respecting the same v-tree

# The Equivalence

**Theorem**

*BCT constraints are polynomially equivalent to SDNNF constraints.*

- BCT constraint $c^* = (\mathbf{x}, P)$ can be transformed into an SDNNF representing $c^*$
- SDNNF $D$ representing constraint $c^*$ can be transformed into a BCT encoding $c^*$

# BCT $\longrightarrow$ SDNNF (Idea)

- BCT constraint $c^* = (\mathbf{x}, P)$
  - BCT $P = (\mathbf{z}, C)$
  - Hidden variables $\mathbf{y} = \mathbf{z} \setminus \mathbf{x}$

1. Make the constraint tree rooted
   - Pick any node as the root
   - Directed the edges away from the root
2. Proceed from the leaves to the root
3. For every variable $z \in \mathbf{z}$ and value $a \in dom(z)$, construct SDNNF $D_{z,a}$ representing the constraints below $z$ assuming literal $(z, a)$

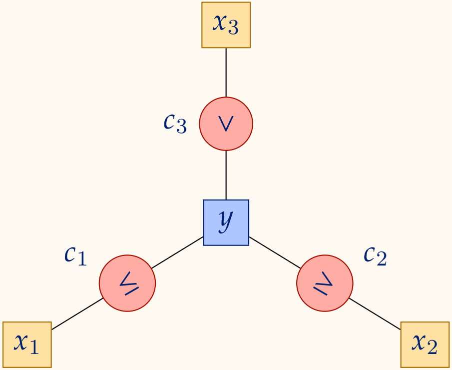
   **Leaf** a single node $(z, a)$

   **Inner** combine the SDNNFs for the constraints "leaving" $z$
4. Construct $D_P$ for the root
   - Combine the SDNNFs for the constraints leaving the root
5. Forget the hidden variables in $D_P$ to obtain $D$ that represents $c^*$

# BCT $\longrightarrow$ SDNNF (Example)

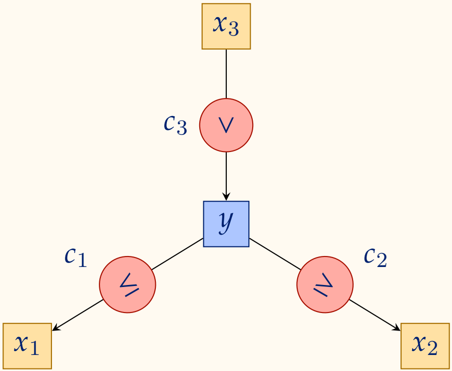
$c^* \equiv \max(x_1, x_2) < x_3$

Domains $\{0, 1, 2\}$



# Rooted Tree

Pick $x_3$ as the root



# SDNNFs For the Leaves

$\boxed{x_1, 0}$ $\boxed{x_1, 1}$ $\boxed{x_1, 2}$

$D_{x_1,0}$ $D_{x_1,1}$ $D_{x_1,2}$

$\boxed{x_2, 0}$ $\boxed{x_2, 1}$ $\boxed{x_2, 2}$

$D_{x_2,0}$ $D_{x_2,1}$ $D_{x_2,2}$

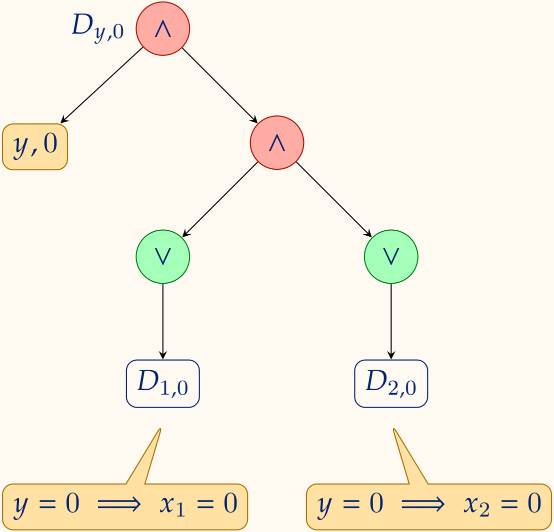
$\boxed{x_1}$

v-tree

$\boxed{x_2}$

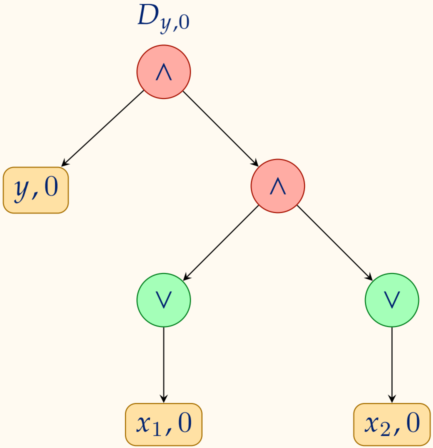
v-tree

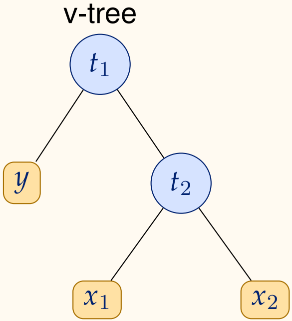
# SDNNF for $(y, 0)$



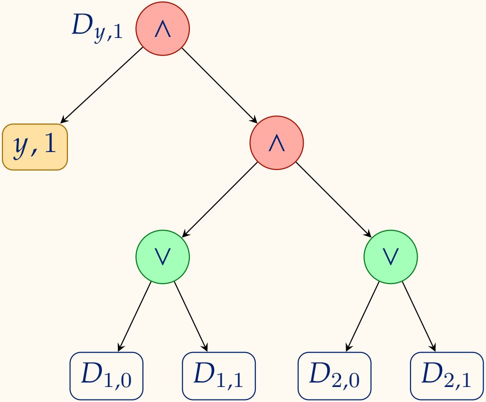
$D_{y,0}$
$\wedge$
$y, 0$
$\wedge$
$\vee$
$\vee$
$D_{1,0}$
$D_{2,0}$
$y = 0 \implies x_1 = 0$
$y = 0 \implies x_2 = 0$

# SDNNF for $(y, 0)$



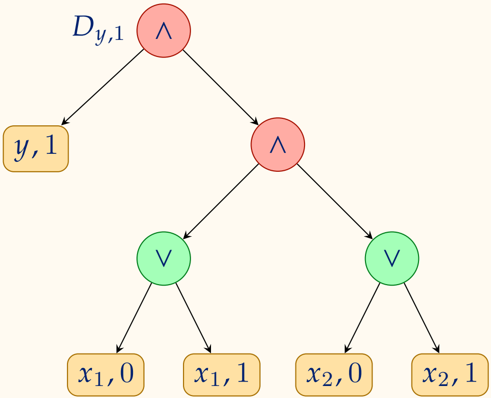
# SDNNF for $(y, 1)$



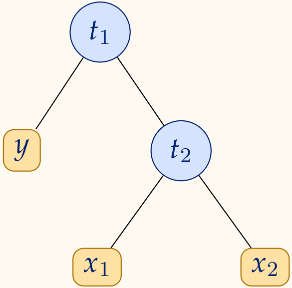

$y = 1 \implies x_1 \leq 1$

$y = 1 \implies x_2 \leq 1$

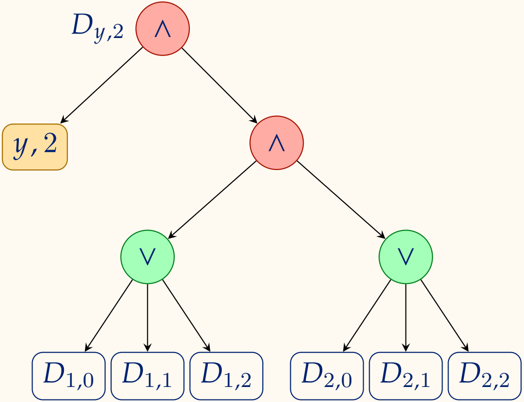
# SDNNF for $(y, 1)$




v-tree

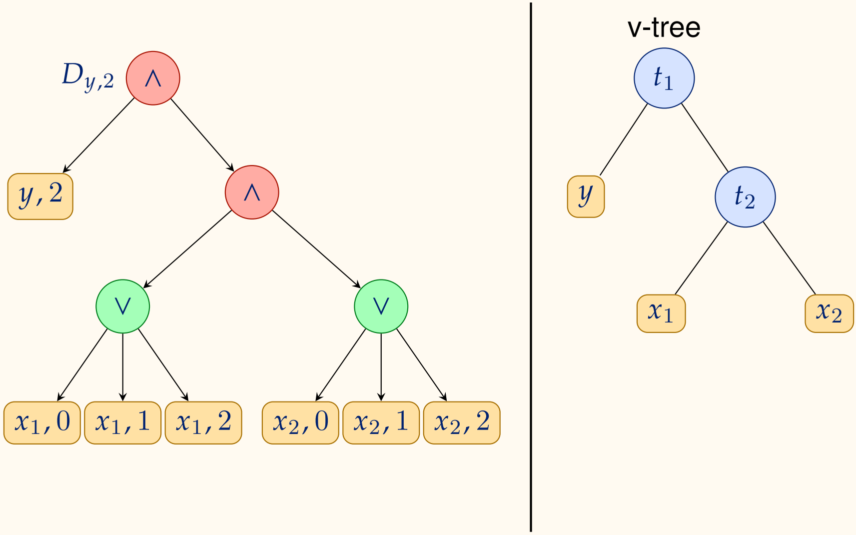


# SDNNF for $(y, 2)$

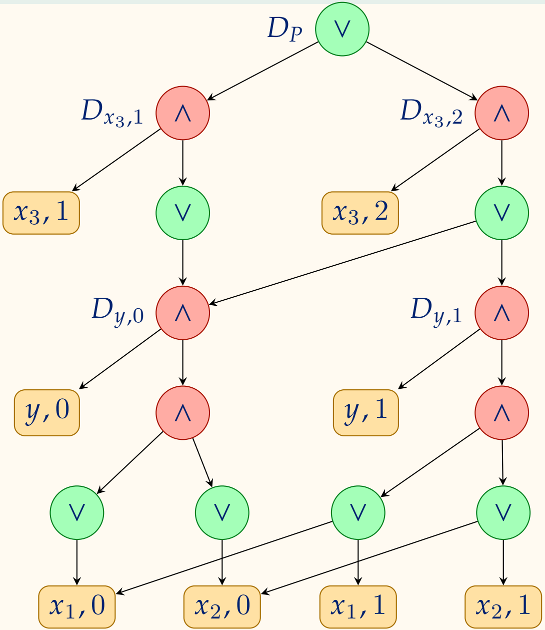



$\boxed{y = 2 \implies x_1 \leq 2}$ $\boxed{y = 2 \implies x_2 \leq 2}$

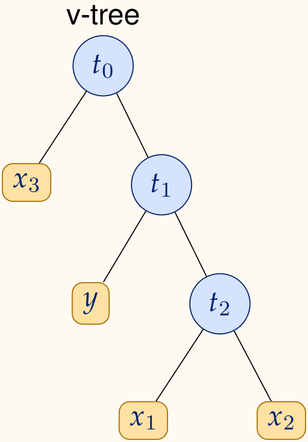
# SDNNF for $(y, 2)$



# SDNNF $D_P$

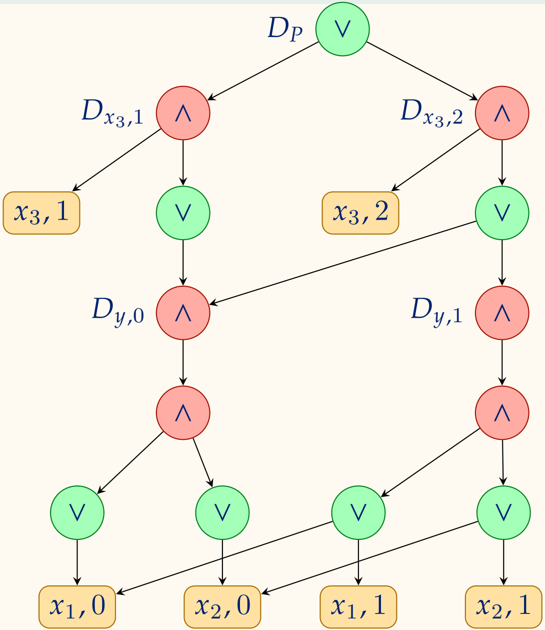


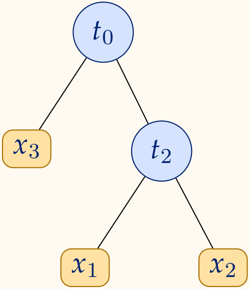
$D_P$
$\vee$
$D_{x_3,1}$
$\wedge$
$D_{x_3,2}$
$\wedge$
$x_3, 1$
$\vee$
$x_3, 2$
$\vee$
$D_{y,0}$
$\wedge$
$D_{y,1}$
$\wedge$
$y, 0$
$\wedge$
$y, 1$
$\wedge$
$\vee$
$\vee$
$\vee$
$\vee$
$x_1, 0$
$x_2, 0$
$x_1, 1$
$x_2, 1$

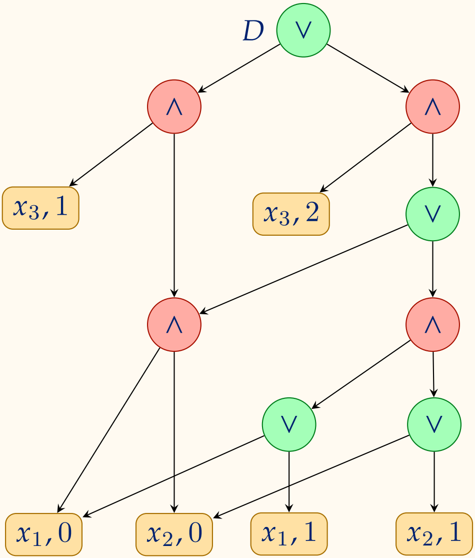
v-tree
$t_0$
$x_3$
$t_1$
$y$
$t_2$
$x_1$
$x_2$

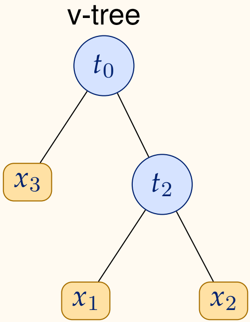
# Forget $y$


$D_P$
$\vee$
$D_{x_3,1}$
$\wedge$
$D_{x_3,2}$
$\wedge$
$x_3, 1$
$\vee$
$x_3, 2$
$\vee$
$D_{y,0}$
$\wedge$
$D_{y,1}$
$\wedge$
$\wedge$
$\wedge$
$\vee$
$\vee$
$\vee$
$\vee$
$x_1, 0$
$x_2, 0$
$x_1, 1$
$x_2, 1$


v-tree
$t_0$
$x_3$
$t_2$
$x_1$
$x_2$

# Simplify


$D$
$\vee$
$\wedge$
$\wedge$
$x_3, 1$
$x_3, 2$
$\vee$
$\wedge$
$\wedge$
$\vee$
$\vee$
$x_1, 0$
$x_2, 0$
$x_1, 1$
$x_2, 1$


v-tree
$t_0$
$x_3$
$t_2$
$x_1$
$x_2$

# SDNNF $\longrightarrow$ BCT (Idea)

**Given**

- SDNNF $D$ representing constraint $c^*$ with scope $\mathbf{x}$
- v-tree $T$

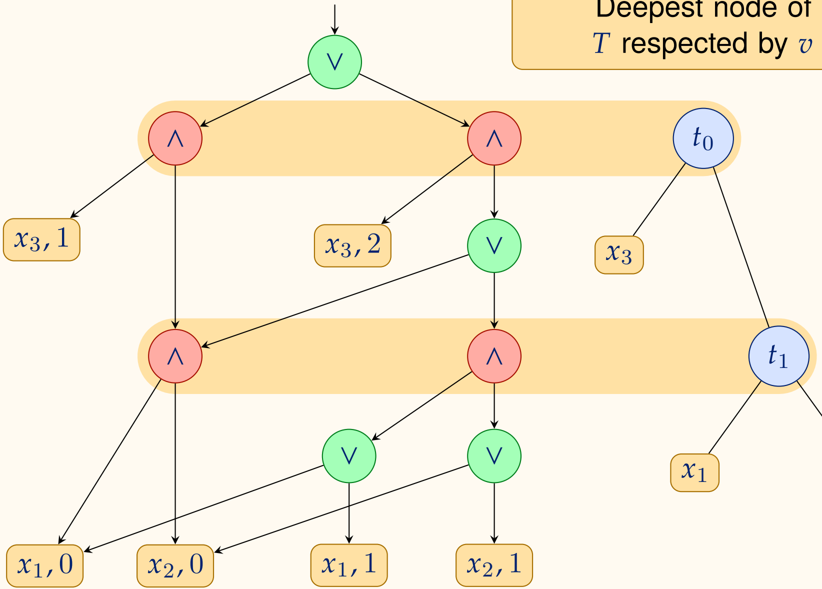
**Construct**

- BCT $P = (\mathbf{z}, C)$ encoding $c^*$
- Structure of $P$ is equal to $T$
- $\mathbf{z} = \mathbf{x} \cup \mathbf{y}$
- Inner node $t$ has an associated hidden variable $y_t \in \mathbf{y}$
- $dom(y_t)$ consists of the $\wedge$ gates with d-node $t$

# D-Node

**d-node of an $\wedge$ gate**

Deepest node of $T$ respected by $v$



# Encoding Certificates

- Certificate $S$ of $D$ = minimal satisfied subtree
- For every node $t$ of the v-tree, $S$ contains exactly one $\wedge$-gate with d-node $t$
  - *Assuming smoothness*
- Relation of constraint $c_{t,t'}$ corresponding to edge $(t, t')$ of $T$:

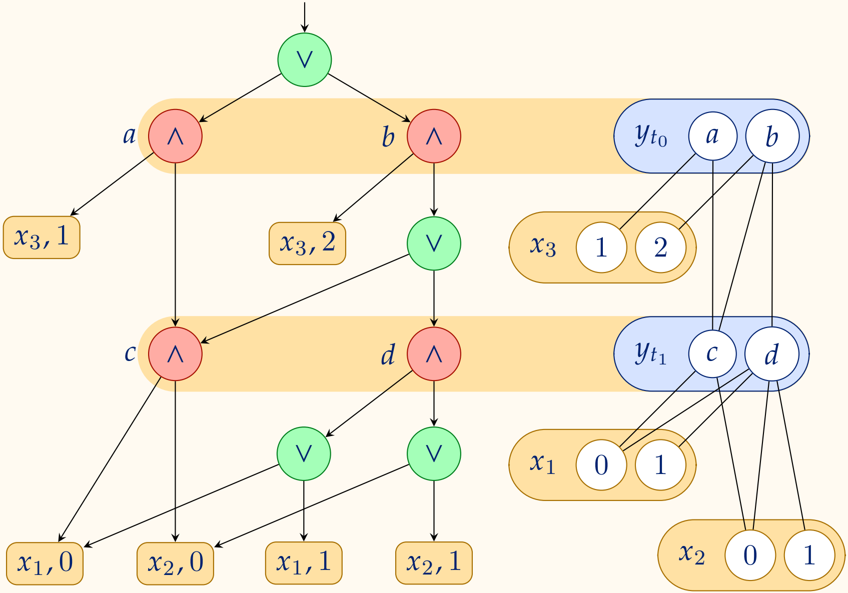
$t'$ is leaf $x \in \mathbf{x}$: pairs $(v, (x, a))$

- $v$ is $\wedge$ gate with d-node $t$
- $(x, a)$ is reachable from $v$ only by $\vee$ gates

$t'$ is an inner node: pairs $(v, v')$

- $v$ is an $\wedge$ gate with d-node $t$
- $v'$ is an $\wedge$ gate with d-node $t'$
- $v'$ is reachable from $v$ only by $\vee$ gates

# Example



# Conclusion

- Construction of an SDNNF from BCT enforces arc consistency
- Size of the SDNNF can be parameterized by the domain size and the treewidth of a binary CSP
  - CSPs can be binarized
  - SDNNF can be constructed for any CSP
- SDNNF restructuring by picking another v-tree node as the root
- All that is known about SDNNFs can be applied to BCTs and vice versa
- Knowledge compilers for compiling into an SDNNF or SDD, can be used to compile into a BCT

# References I


Darwiche, Adnan (1999). "Compiling Knowledge into Decomposable Negation Normal Form". In: *Proceedings of the 16th International Joint Conference on Artifical Intelligence - Volume 1*. IJCAI'99. Stockholm, Sweden: Morgan Kaufmann Publishers Inc., pp. 284–289.

Gange, Graeme and Peter J. Stuckey (2012). "Explaining Propagators for s-DNNF Circuits". In: *Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimization Problems*. Ed. by Nicolas Beldiceanu, Narendra Jussien, and Éric Pinson. Springer Berlin Heidelberg, pp. 195–210. ISBN: 978-3-642-29828-8.

# References II


- Pipatsrisawat, Knot and Adnan Darwiche (2008). “New Compilation Languages Based on Structured Decomposability”. In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 1*. AAAI’08. Chicago, Illinois: AAAI Press, pp. 517–522. ISBN: 978-1-57735-368-3.
- Wang, Ruiwei and Roland H. C. Yap (2022a). “CNF Encodings of Binary Constraint Trees”. In: *28th International Conference on Principles and Practice of Constraint Programming (CP 2022)*. Ed. by Christine Solnon. Vol. 235. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 40:1–40:19. ISBN: 9783959772402. DOI: `10.4230/LIPIcs.CP.2022.40`.

# References III


- Wang, Ruiwei and Roland H. C. Yap (June 2022b). "Encoding Multi-Valued Decision Diagram Constraints as Binary Constraint Trees". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.4, pp. 3850–3858. DOI: 10.1609/aaai.v36i4.20300.
- — (June 2023). "The Expressive Power of Ad-Hoc Constraints for Modelling CSPs". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 37.4, pp. 4104–4114. DOI: 10.1609/aaai.v37i4.25526.