Soft constraints: Algorithms (3)

T. Schiex

INRA - Toulouse, France

Inference

In classical CSP, inference produces new constraints which are implied by the problem. Makes implicit c explicit.

$\langle X, D, C\rangle \rightarrow c$ s.t. c satisfied by all solutions.

$$K \subset C, L = \cup_{c_S \in K}(S), V \subset L, \quad K \rightarrow c = (\bowtie_{c \in K} c)[V]$$

Then $\langle X, D, C \cup \{c\}\rangle$ is equivalent to $\langle X, D, C\rangle$ (same solutions). More explicit. Simpler to solve.

Incomplete inference: transform $\langle X, D, C\rangle$ into an equivalent problem where all possible local inferences have been performed.

Node/Arc consistency and binary CSP

- Node consistency: the empty assignment can be extended to one variable in a consistent way (unary constraints).
- Arc consistency: each value of each variable can be extended to 2 variables in a consistent way.

  Enforcing by inference on every binary constraint: $c_i = c_i \bowtie (c_{ij} \bowtie c_j)[i]$. Infers all unary constraints implied by c_{ij}.

Local consistency:Polynomial time, yields a unique equivalent, more explicit problem that satisfies the property.

Soft constraints

$P = \langle X, D, C, S\rangle$ describes a distribution $P(t)$ of valuations on the search space (combination of all constraints).

We say that c_s is implied by P iff $\forall t, c_S(t[S]) \succcurlyeq_s P(t)$.

$$K \subset C, L = \cup c_S \in K(S), V \subset L, \quad K \rightarrow (\bowtie_{c \in K} c)[V]$$

Adding c_S to P may change the distribution of valuations unless. . . $\oplus$ idempotent.

Local consistency for idempotent SCSP

Consider a binary SCSP $\langle X, D, C, S \rangle$.
$C = \{c_\varnothing\} \cup C^1 \cup C^+$.

A CSP is node-consistent iff $c_\varnothing$ implies any $c_i[]$ (nothing more to infer).

A variable i is arc consistent wrt $c_{ij} \in C$ iff c_i implies $c_{ij}[i]$.

The soft CN is arc-consistent iff all its variables are AC wrt. all constraints it involves.

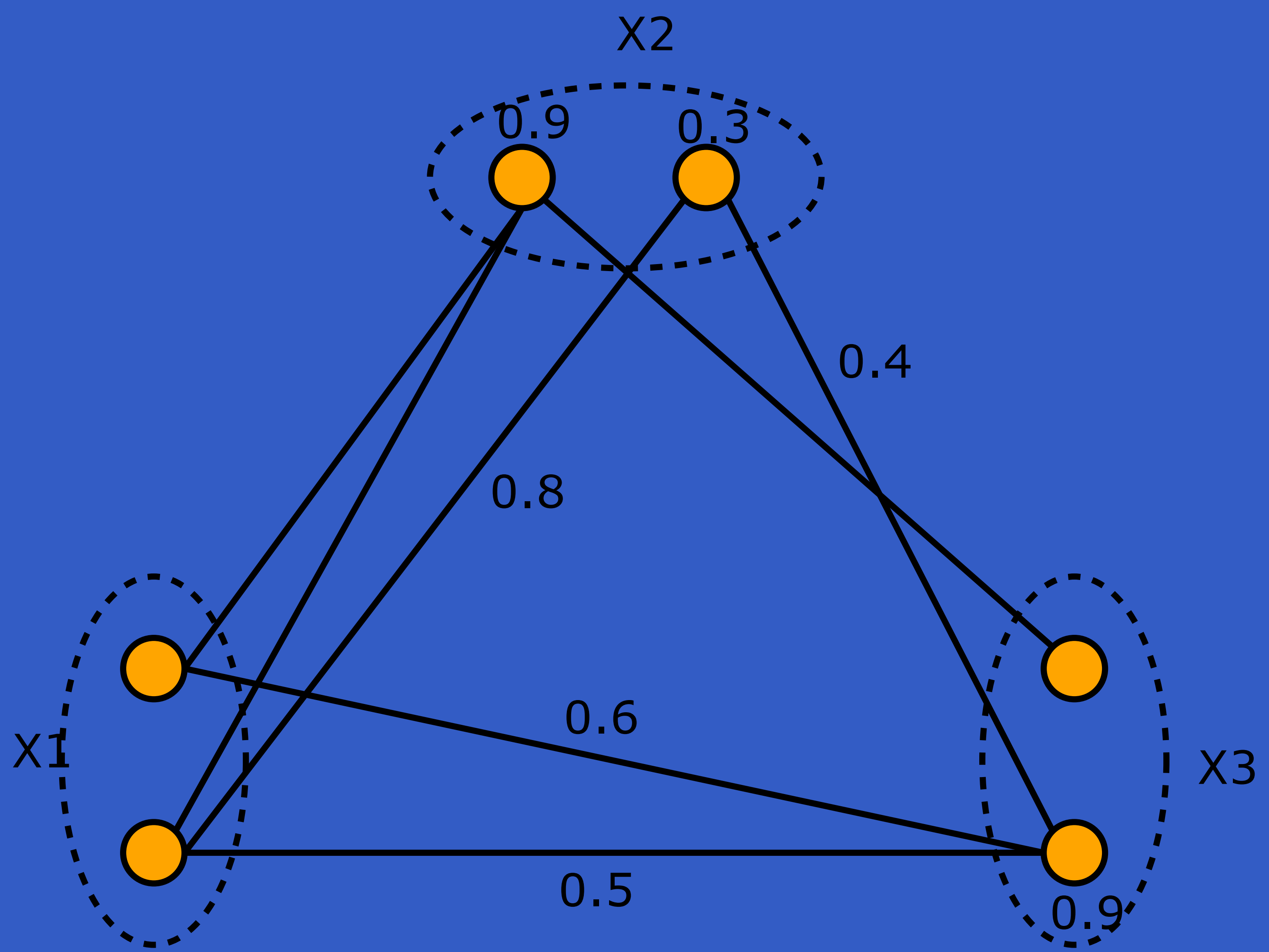
Enforcing AC on idempotent SCSP

For all $x_i \in X$, $c_{ij} \in C$, do

$$c_i = c_i \bowtie (c_j \bowtie c_{ij})[i]$$

until fixpoint. Can be more expensive that in classical case (still polynomial). Limited variable elimination w/o elimination.

Example on a fuzzy CN



k-consistency

Consider $W \subset X, |W| = k - 1$. Let $C(V)$ be the constraints whose scope is included in V.

W is k-consistent iff

$$\forall x \in X \setminus W, \bowtie C(W) \to (\bowtie C(W \cup \{x\}))[W]$$

A CN is k-consistent iff all subsets of size $k - 1$ are k-consistent.

Enforcing k-consistency

For all $W \subset X, |W| = k - 1$ and for all $x \in X \setminus W$ do:

$$c_W = c_W \bowtie C(W \cup \{x\})[W]$$

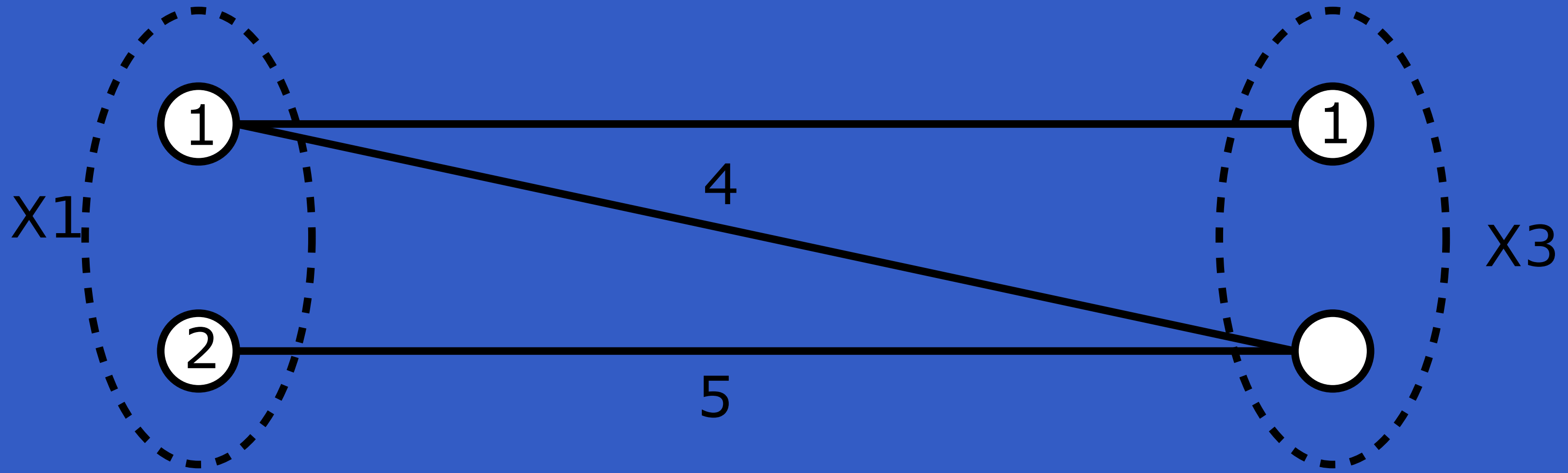
until fixpoint.

- Generates arity $k - 1$ constraints.
- Time exponential in k, space exp. in $k - 1$.
- $|X|$-consistency infers more constraints than VE or BBE and makes all implied constraints explicit.

Non idempotent VCSP: additive CSP

It does not work...

$$S = \langle \mathbb{N} \cup \{\infty\}, <, +, \perp = 0, \top = \infty \rangle$$



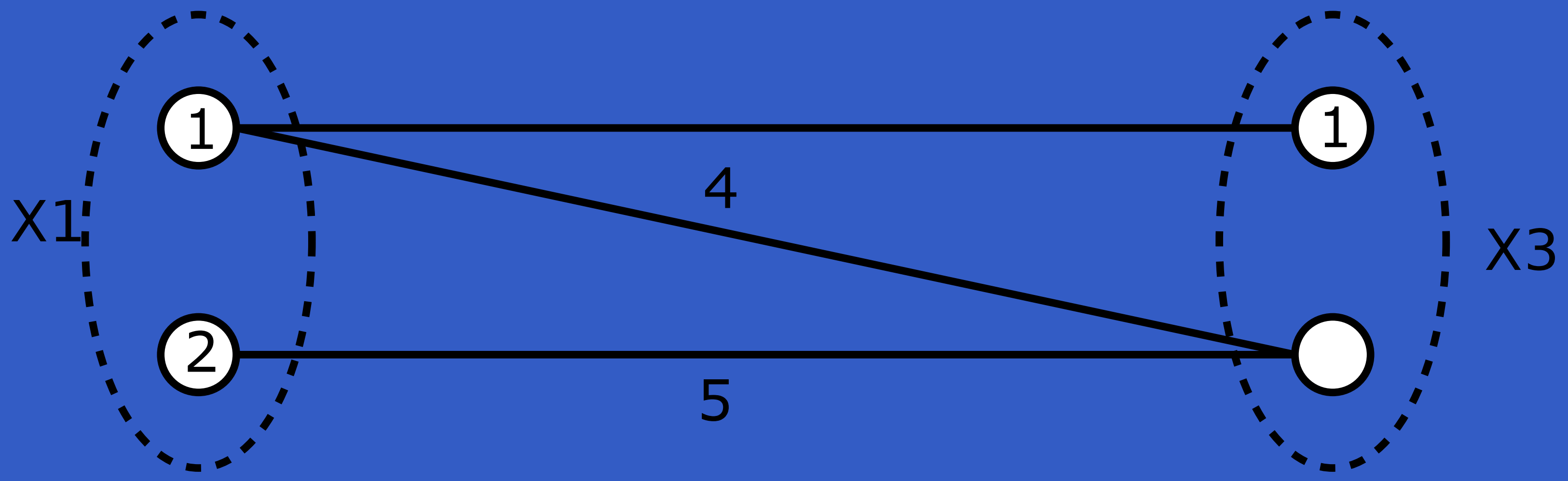
What can be done ?

We don’t want to eliminate. We cannot add implied constraints…

- when we project some penalty out of a constraint to a variable and add it to the problem
- we must compensate for this by “substracting” it from the constraint

Non idempotent VCSP: additive CSP

$$S = \langle \mathbb{N} \cup \{\infty\}, <, +, \bot = 0, \top = \infty \rangle$$



Fair VCSPs

In a valuation structure $S = \langle E, \oplus, \succcurlyeq \rangle$, if $\alpha, \beta \in E$, $\alpha \preccurlyeq \beta$ and there exists a valuation $\gamma \in E$ such that $\alpha \oplus \gamma = \beta$, then γ is known as a difference of β and α.

The valuation structure S is fair if for any pair of valuations $\alpha, \beta \in E$, with $\alpha \preccurlyeq \beta$, there exists a maximum difference of β and α. This unique maximum difference of β and α is denoted by $\beta \ominus \alpha$.

What valuation structures are fair ?

- Classical CSP: $\ominus = \max$
- Possibilistic (min-max) CSP: $\ominus = \max$
- Weighted CSP (min-+) CSP: $\ominus = -$
- Probabilistic CSP: $\ominus = \div$

Lexicographic CSP can be turned in to weighted CSP or the structure modified so that $\ominus$ exists.

Not fair ?

$S = \langle \mathbb{N} \cup \{\infty, \top\}, \geq, \oplus, \bot, \top \rangle$

- n year of prison (finite)
- life imprisonment (∞)
- death penalty ($\top$).

Two life sentences $\rightarrow$ death sentence (($\infty \oplus \infty) = \top$).

$\forall m, n \in \mathbb{N}, (m \oplus n = m + n)$; $\forall n \in \mathbb{N}, (\infty + n = \infty)$;
$\forall \alpha \in E, (\top \oplus \alpha = \top)$.

Not fair: differences exist. Set of differences of ∞ and ∞ is $\mathbb{N}$. No maximum difference.

Binary weighted CSP

Binary additive CSP with... an upper bound k.

$S(k) = \langle [0,k], \leq, \oplus, 0, k \rangle$. $<$ usual order on integers.

$$a \oplus b = \min(k, a+b)$$

$$a \ominus b = \begin{cases} a - b & : \quad a \neq k \\ k & : \quad a = k \end{cases}$$

Projecting and preserving equivalence

Let $\alpha = \min_{b \in D_j}(c_{ij}(a, b))$.

Procedure `Project` (i, a, j, α)
$c_i(a) := c_i(a) \oplus \alpha$;
foreach $b \in D_j$ **do** $c_{ij}(a, b) := c_{ij}(a, b) \ominus \alpha$;

Information flows from c_{ij} to c_i. Preserves solutions.

Another "equivalence preserving" op.

Let $\beta = c_i(a)$.

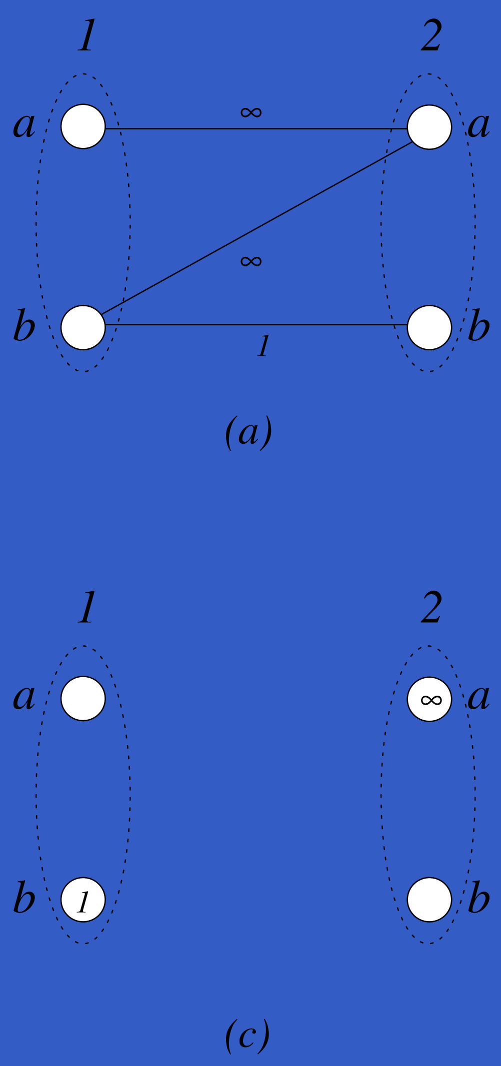
Procedure $\mathit{Extend}(i, a, j, \beta)$

foreach $b \in D_j$ **do** $c_{ij}(a,b) := c_{ij}(a,b) \oplus \beta$;

$c_i(a) := c_i(a) \ominus \beta$;

Information flows from $c_i(a)$ to $c_{ij}(a,b)$. Preserves solutions.

Let’s play



Node Consistency

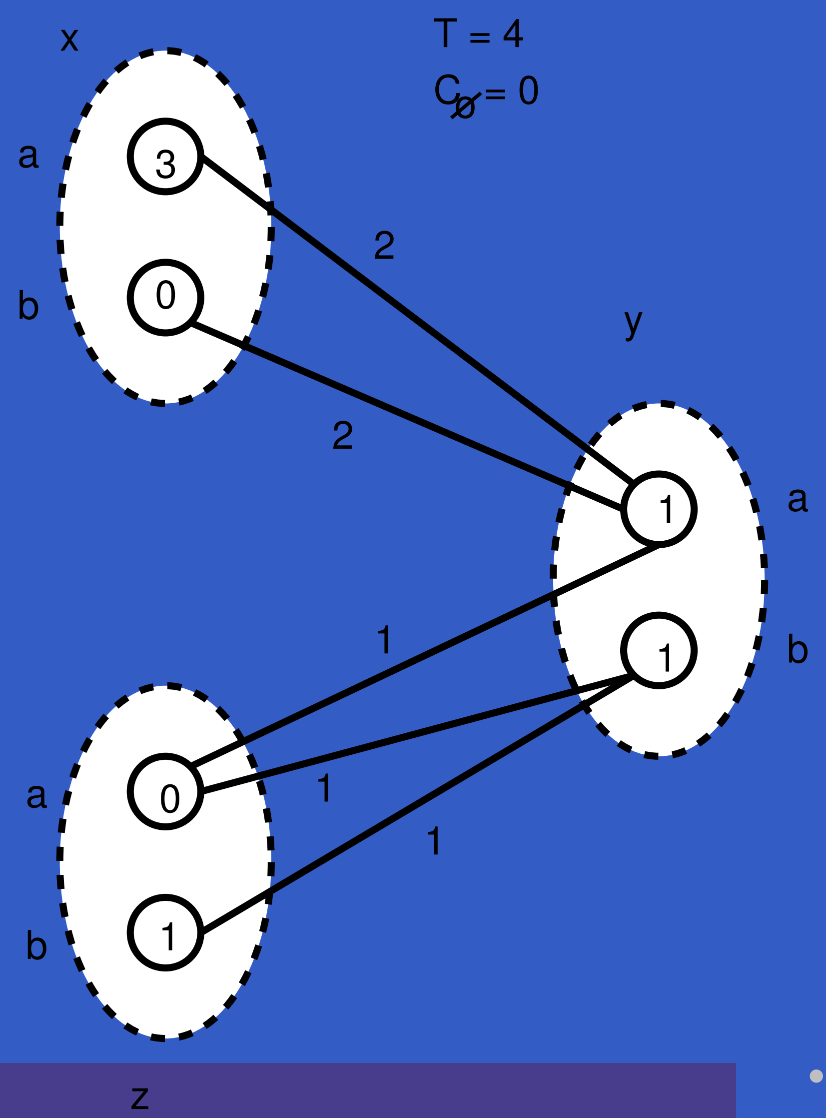
Node consistency: all possible information in the c_i has been extracted by `Project` to $c_\varnothing$.

$\forall i \in X$

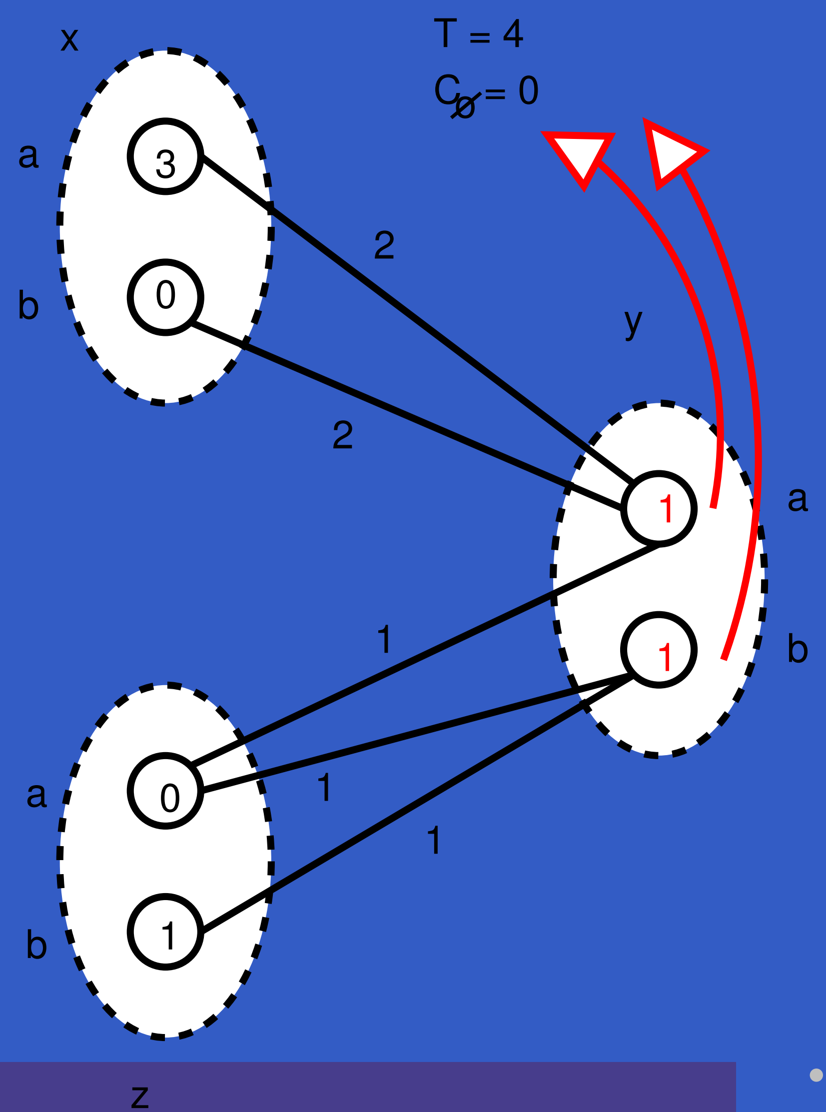
- $\exists a \in D_i, c_i(a) = 0$ (support for $c_\varnothing$).
- $\forall a \in D_i, c_\varnothing \oplus c_i(a) \prec \top$

Can delete $a \in D_i$ whenever $c_\varnothing + c_i(a) = k$.

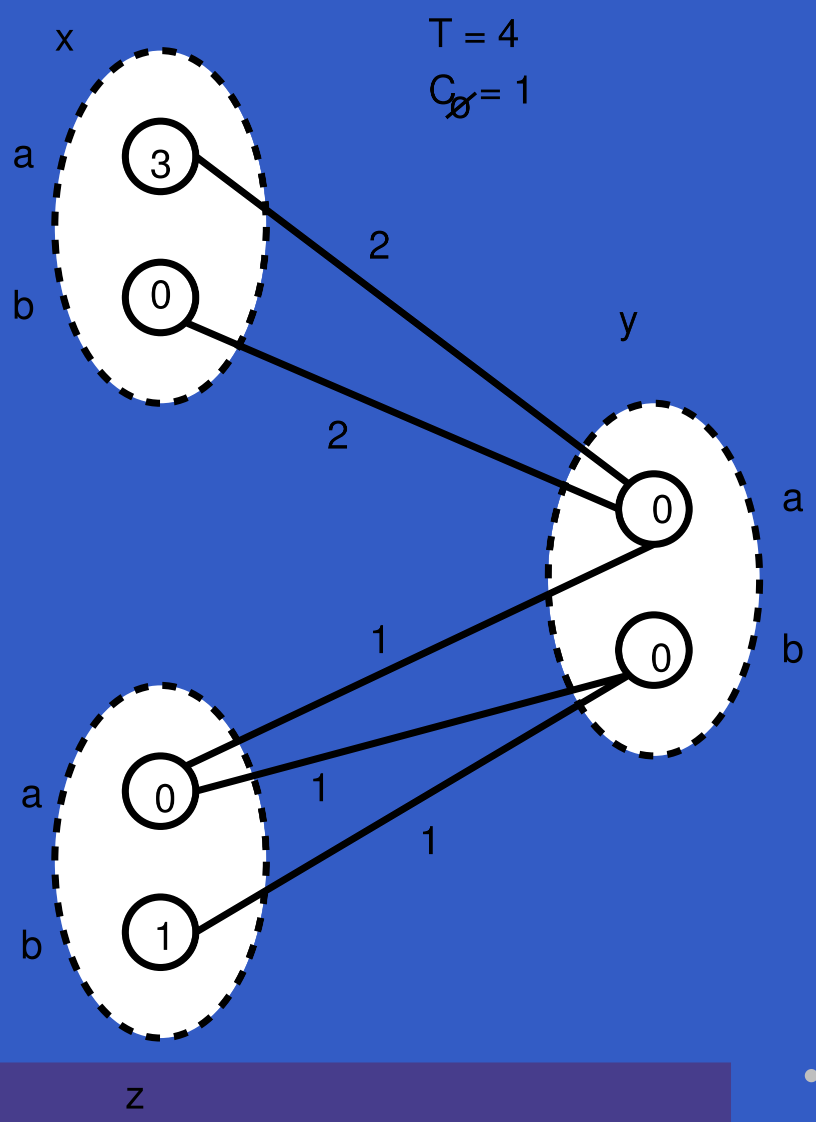
NC in action



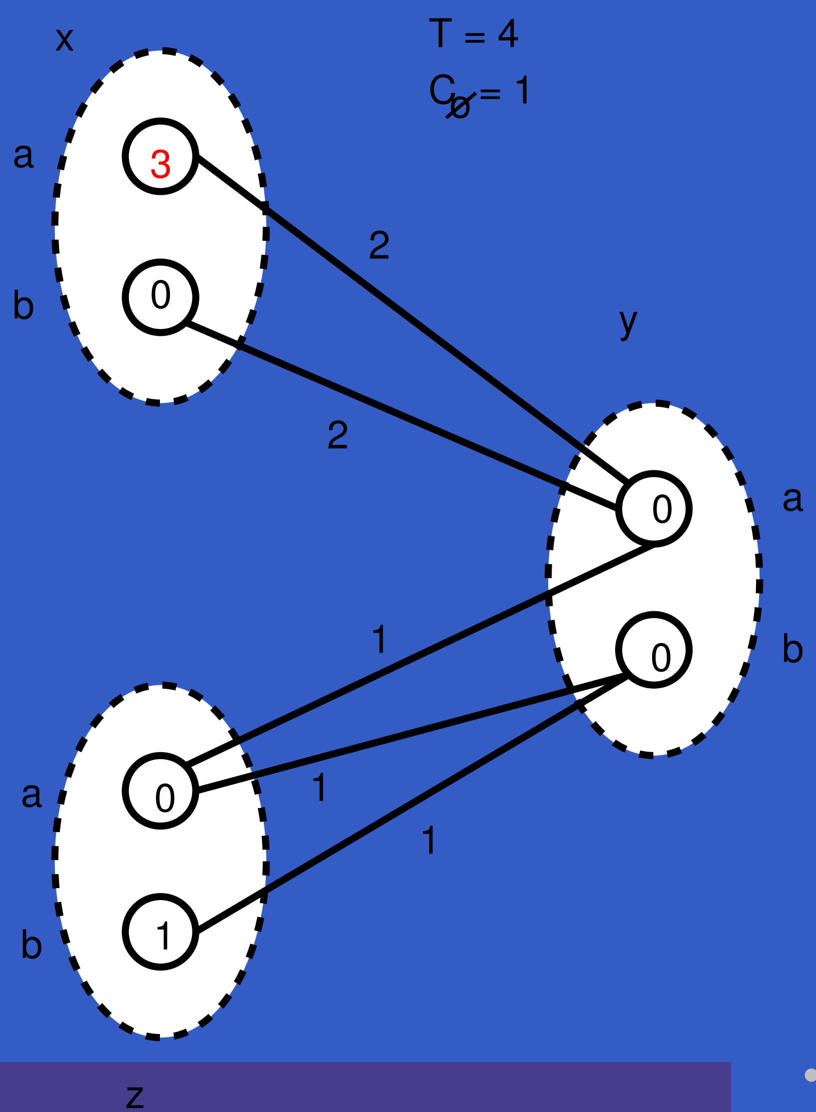
NC in action



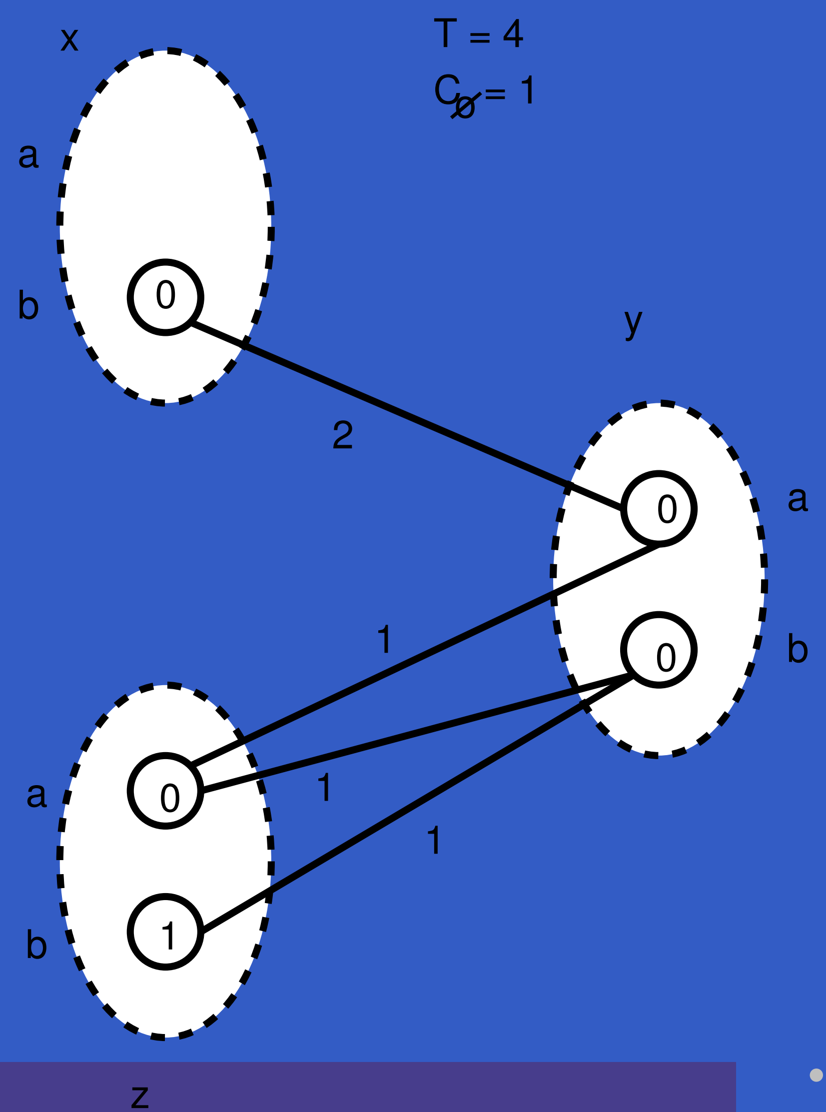
NC in action



NC in action



NC in action



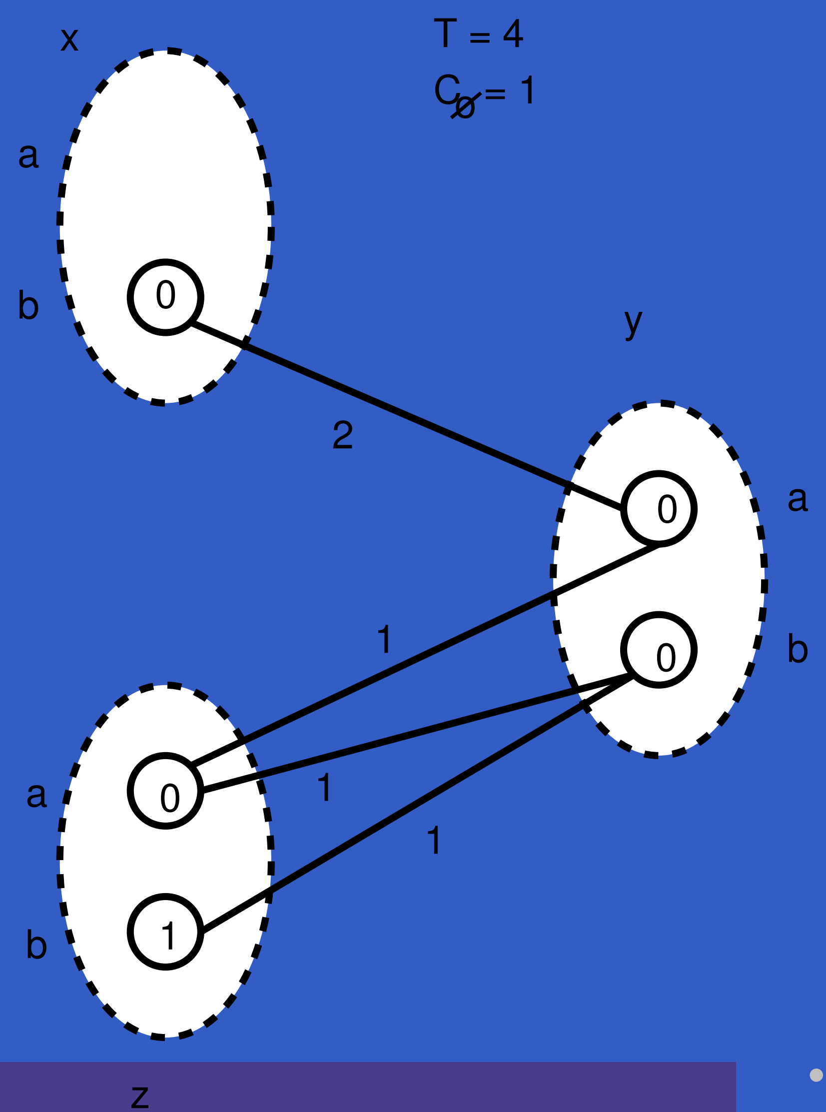
Arc consistency

Arc consistency: all information that can be projected out of all c_{ij} has been projected out.

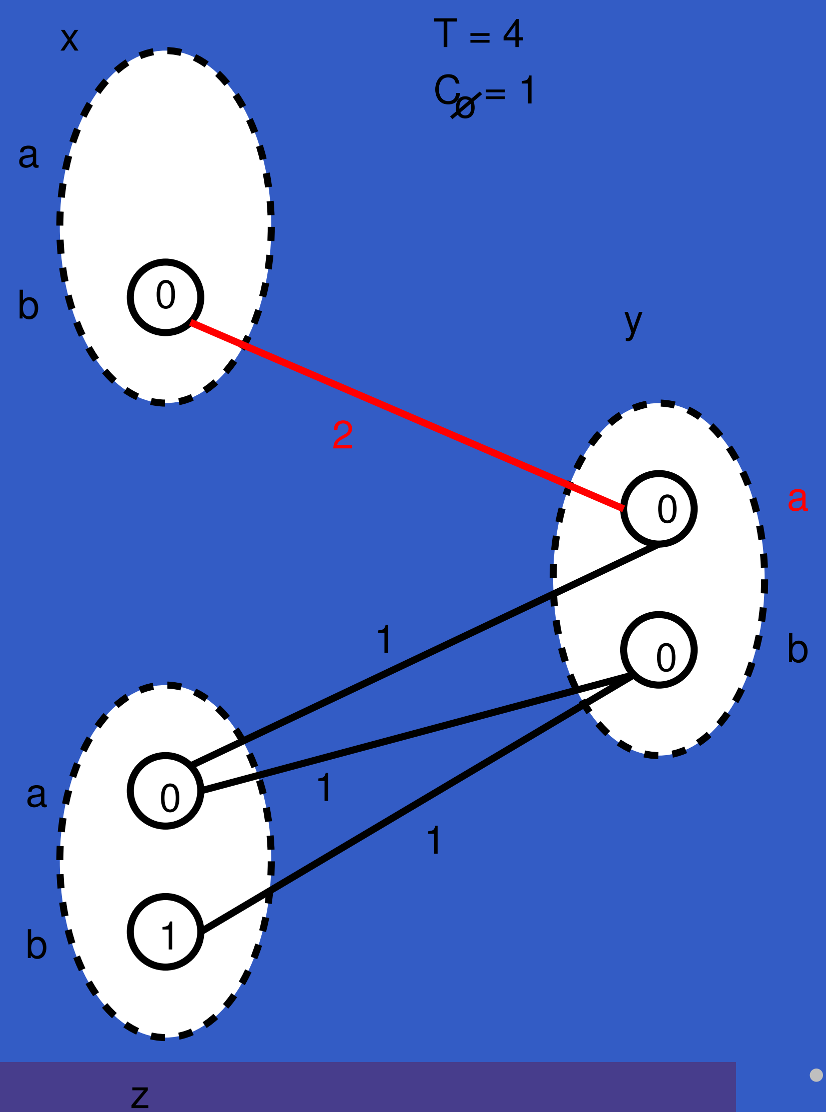
NC and $\forall i, j$ s.t. $c_{ij} \in C$

- $\forall \in D_i \exists b \in D_j$ s.t. $c_{ij}(a, b) = \bot$
- b is a support for a on c_{ij}.

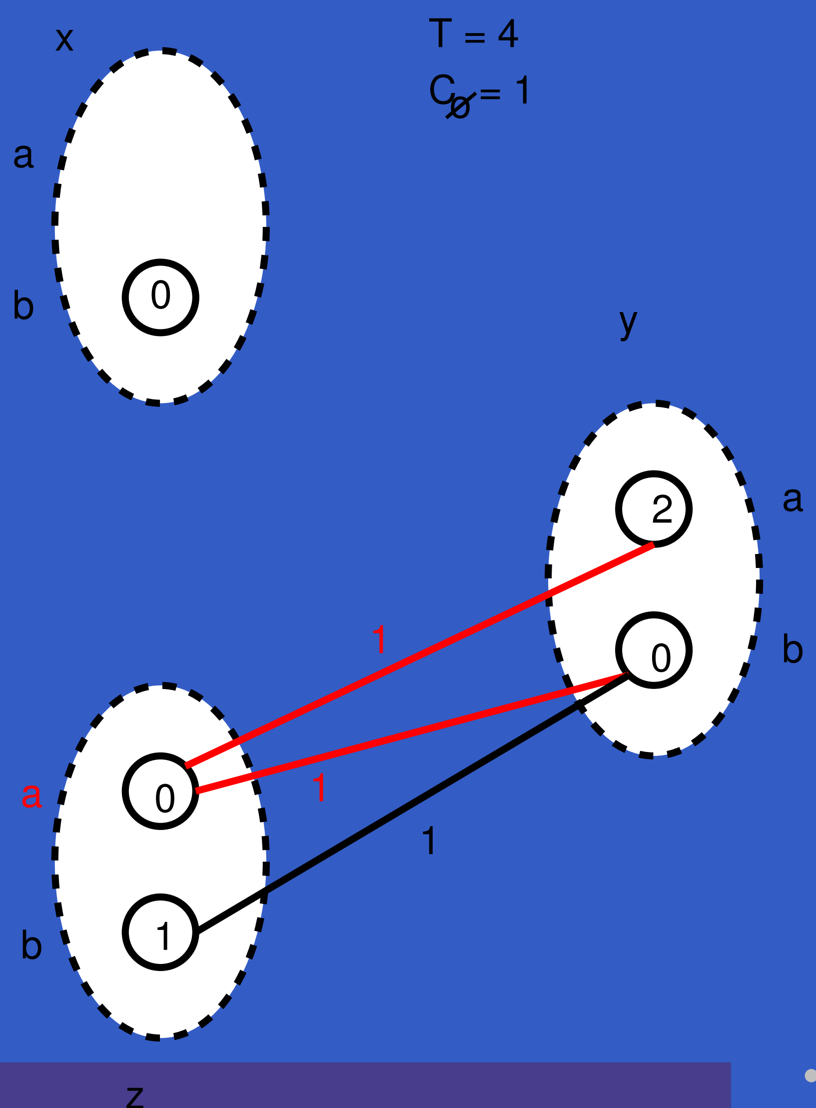
In action



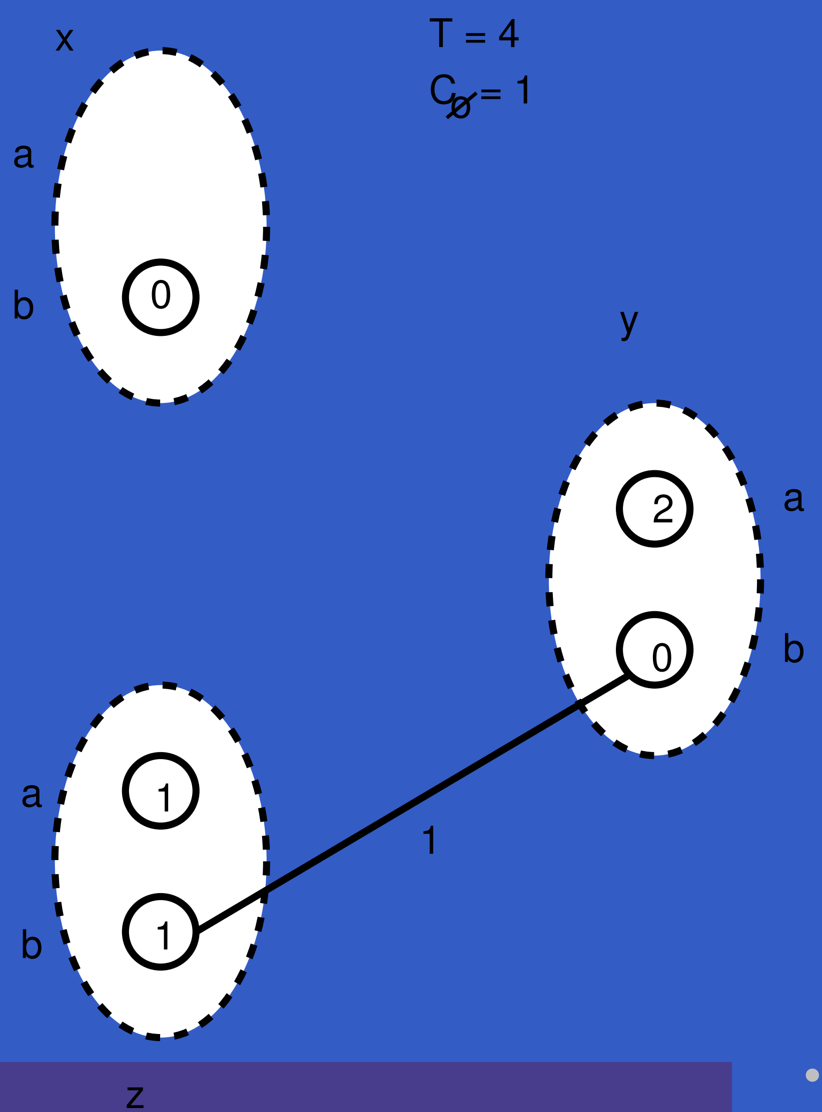
In action



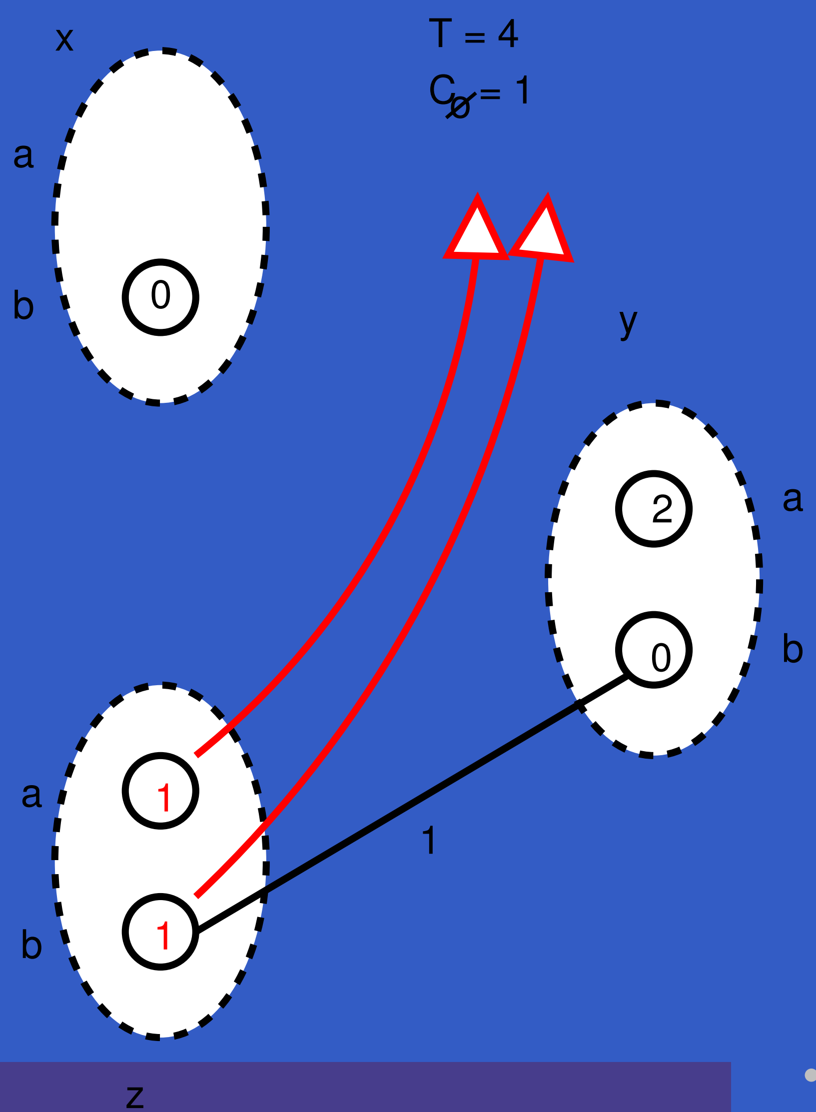
In action



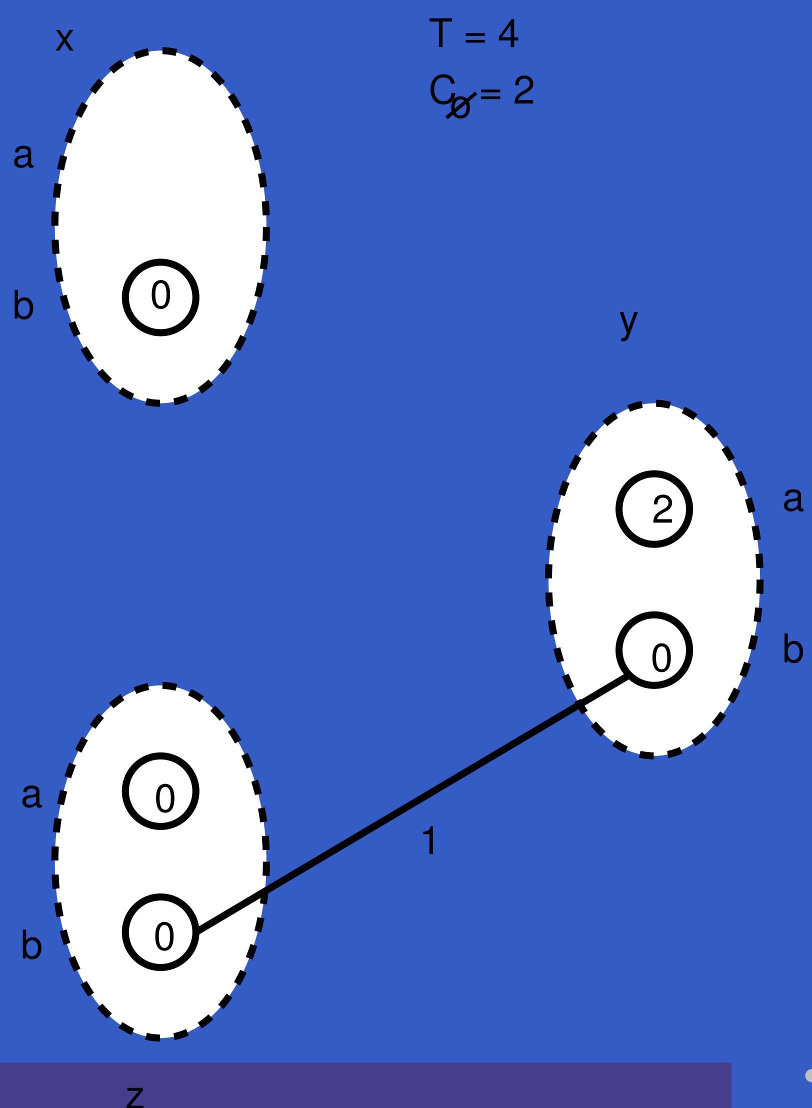
In action



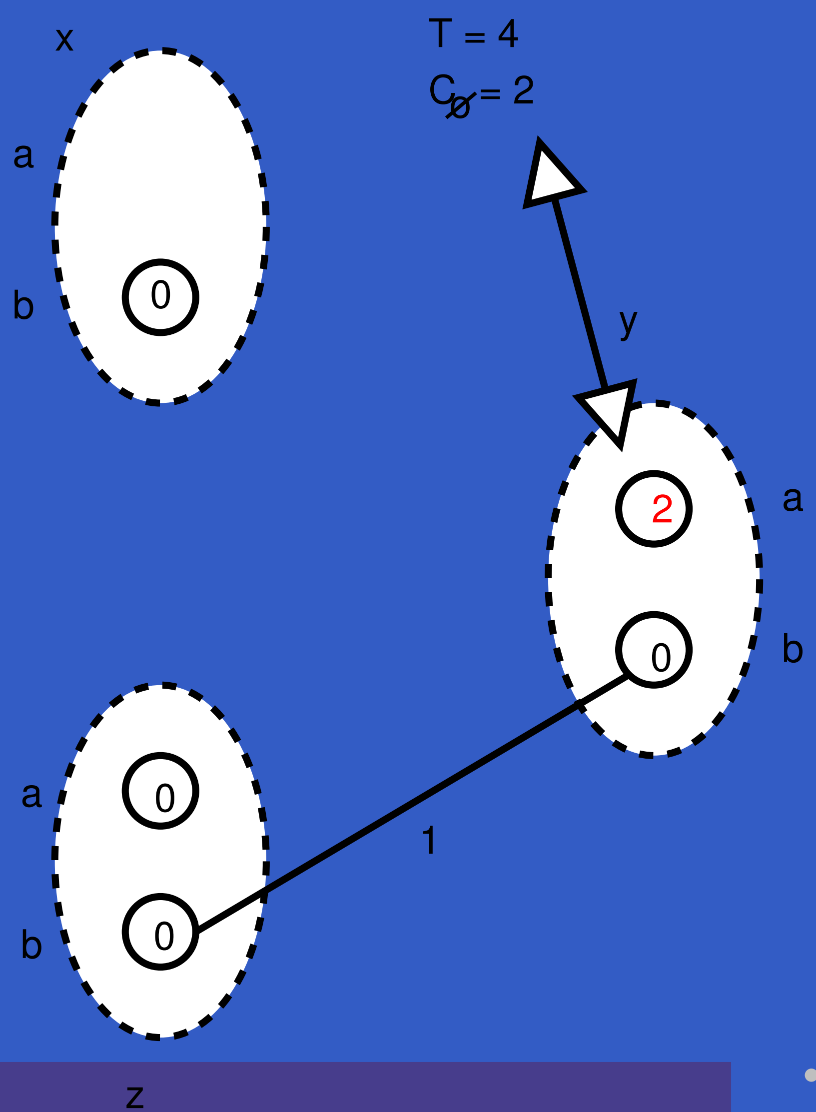
In action



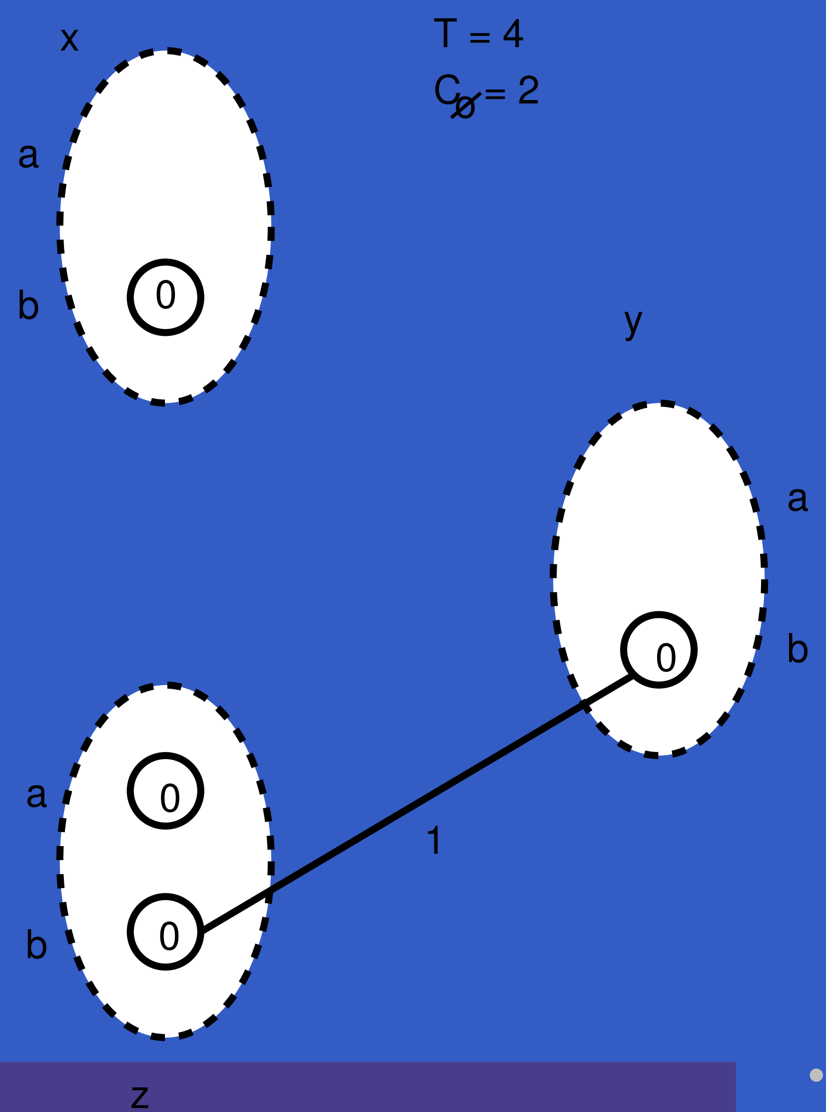
In action



In action



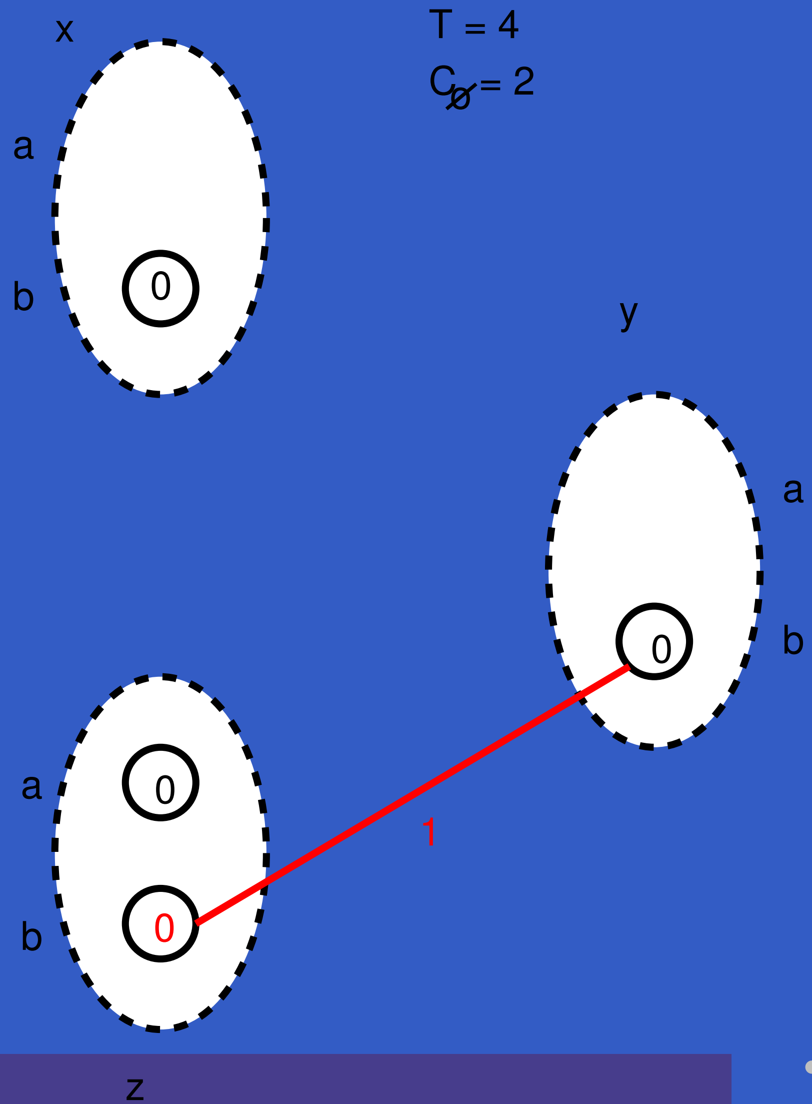
In action



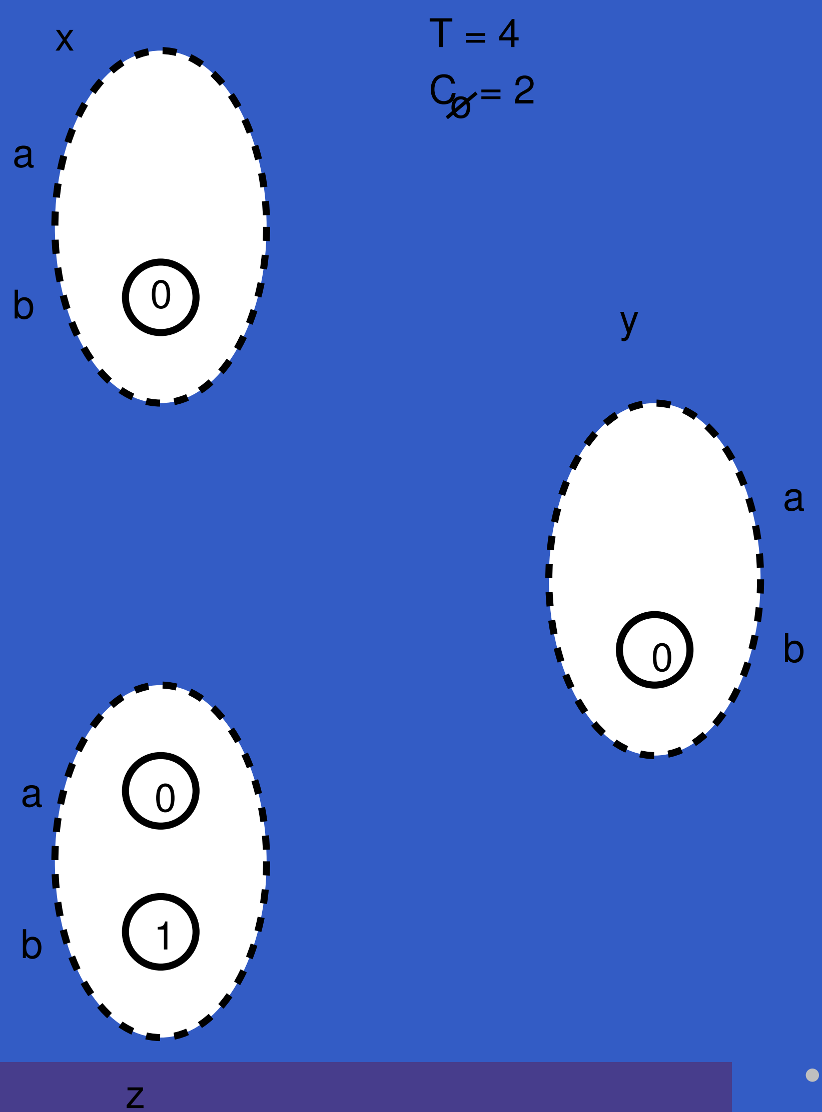
In action

$T = 4$

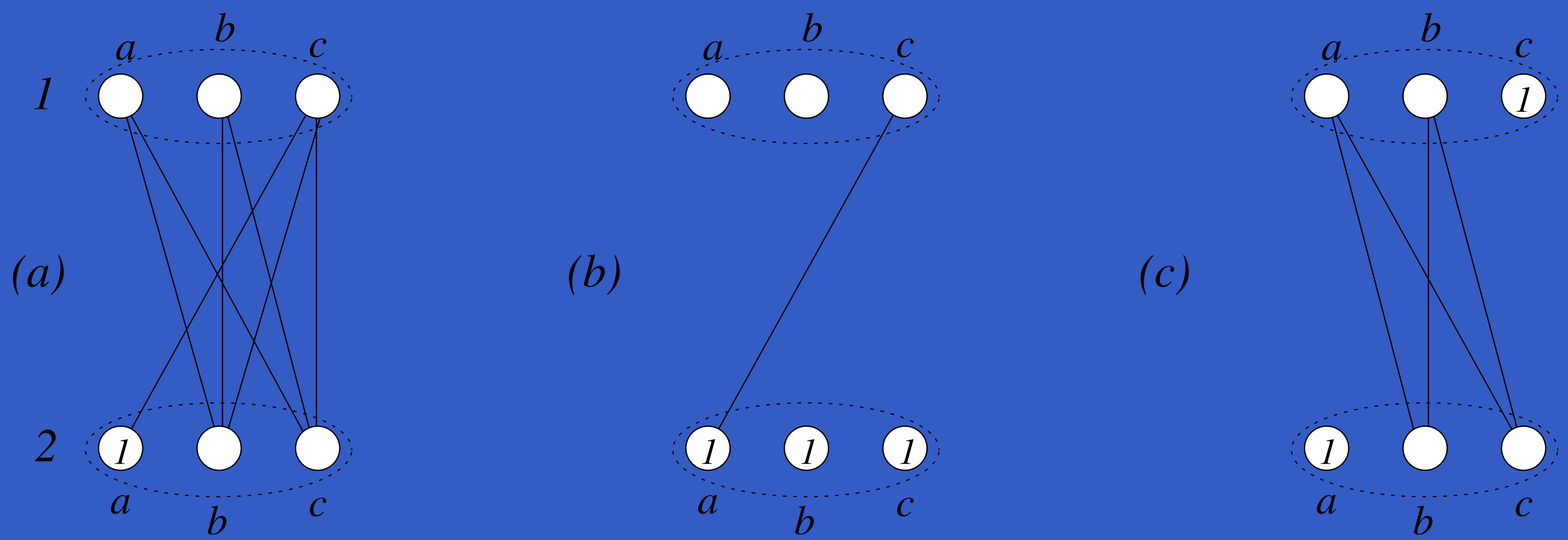
$C_\varnothing = 2$



In action



Loss of uniqueness of the fixpoint



Complexity of AC 2001 based implemen

- a queue Q of variable to process (pruned domains)
- $S(i,a,j)$: current support for (i,a)
- $S(i)$: current support for i on $C_\varnothing$

AC 2001 based

Procedure `ProjectUnary` (i)

- $S(i) := argmin_{a \in D_i}\{c_i(a)\}$;
- $\alpha := c_i(S(i))$;
- $c_\varnothing := c_\varnothing \oplus \alpha$;
- **foreach** $a \in D_i$ **do** $c_i(a) := c_i(a) \ominus \alpha$;

Function `FindSupportAC*` (i, j)

- **foreach** $a \in D_i$ **s.t.** $S(i,a,j) \notin D_j$ **do**
  - $S(i,a,j) := argmin_{b \in D_j}\{c_{ij}(a,b)\}$;
  - $\alpha := c_{ij}(a, S(i,a,j))$;
  - `Project` (i,a,j,α);
- `ProjectUnary` (i);

Soft AC

Function `PruneVar(i)` : *boolean*

change $:=$ **false**;
foreach $a \in D_i$ *s.t.* $(c_i(a) \oplus c_\varnothing = \top)$ **do**
  $D_i := D_i - \{a\}$;
  change $:=$ **true**;
return *change*;

Soft AC

Procedure *AC* ()*

$Q = \{1, \ldots, n\}$;
while $(Q \neq \emptyset)$ **do**
  $j := \mathrm{pop}(Q)$;
  for $c_{ij} \in \mathcal{C}$ **do** `FindSupportAC*` (i, j);
  foreach $i \in \mathcal{X}$ **do**
1   **if** *PruneVar* (i) **then** $Q := Q \cup \{i\}$;

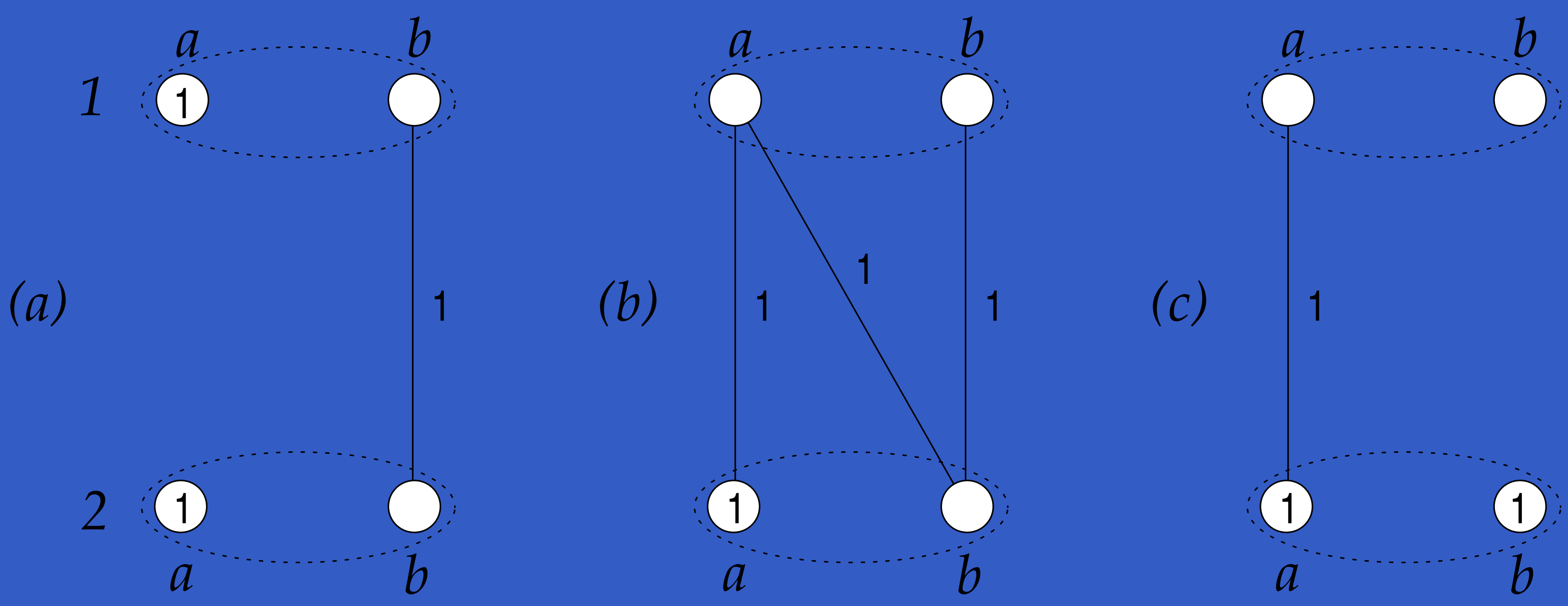
Complexity

- `PruneVar` is $O(d)$ and `FindSupportAC*` is $O(d^2)$
- a var. j is added to Q at most $d+1$ times (at start, each deletion)
- each c_{ij} considered at most $d+1$ times (in each direction): $e(d+1)$ calls to `FindSupportAC*`.
- `AC*` while loop: atmost nd times, $O(n^2d)$ calls to `PruneVar`.

Time $O(n^2d^2 + ed^3)$. Space can be reduced to $O(ed)$.

Can we do more ?

AC: information flows from binary to unary. And the converse ?



And back again ? No fix point !

Directional AC

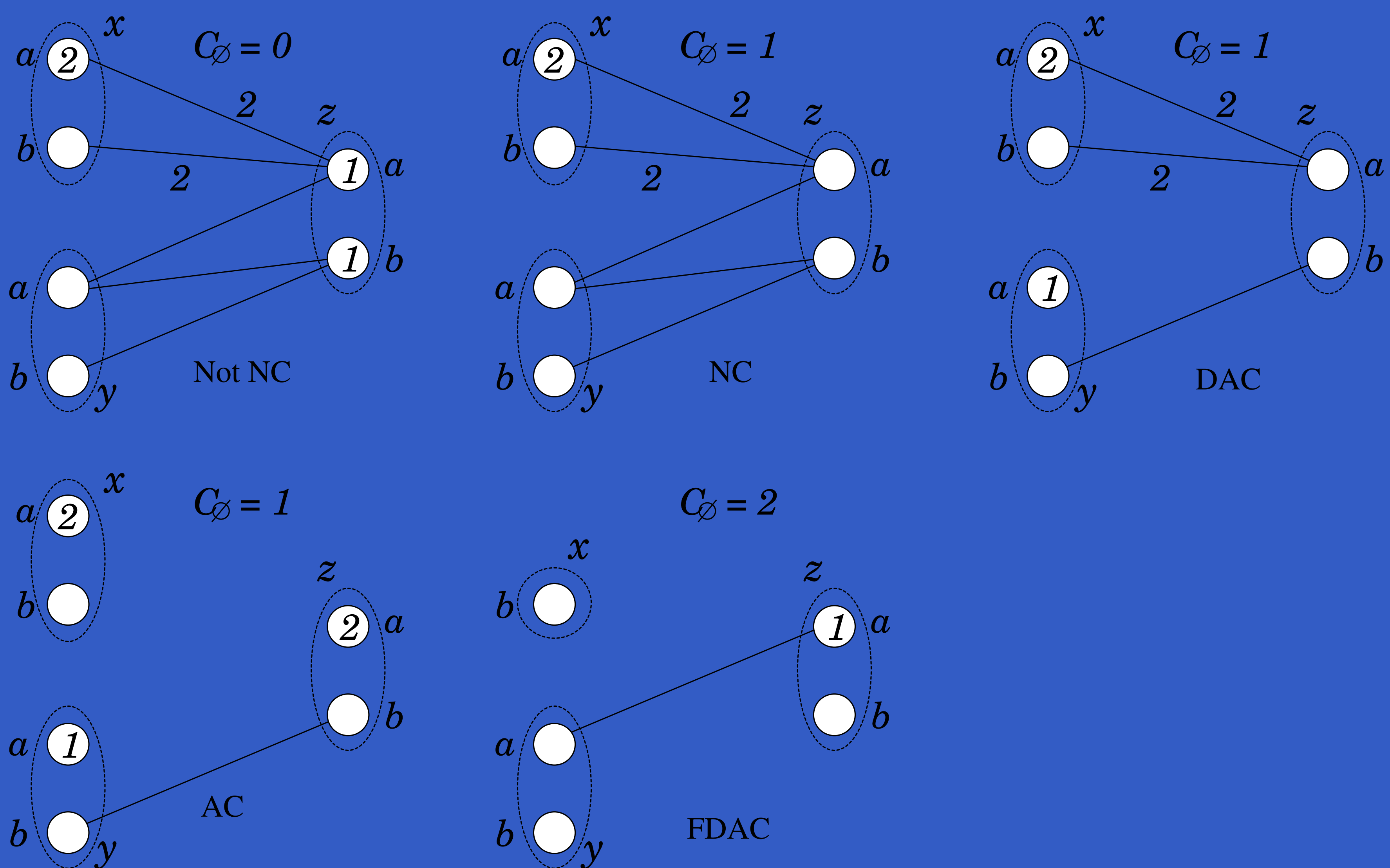
Another way to enforce a fix point: an order on variables $i < j$.

DAC: NC and $\forall i, j$ s.t. $c_{ij} \in C, i < j$

- $\forall \in D_i \exists b \in D_j$ s.t. $c_{ij}(a, b) \oplus c_j(b) = \perp$
- b is a full support for a on c_{ij}.

Full DAC = AC + DAC.

NC, DAC, AC, FDAC (xyz)



Complexities/strengths

Using an AC2001 based propagation.

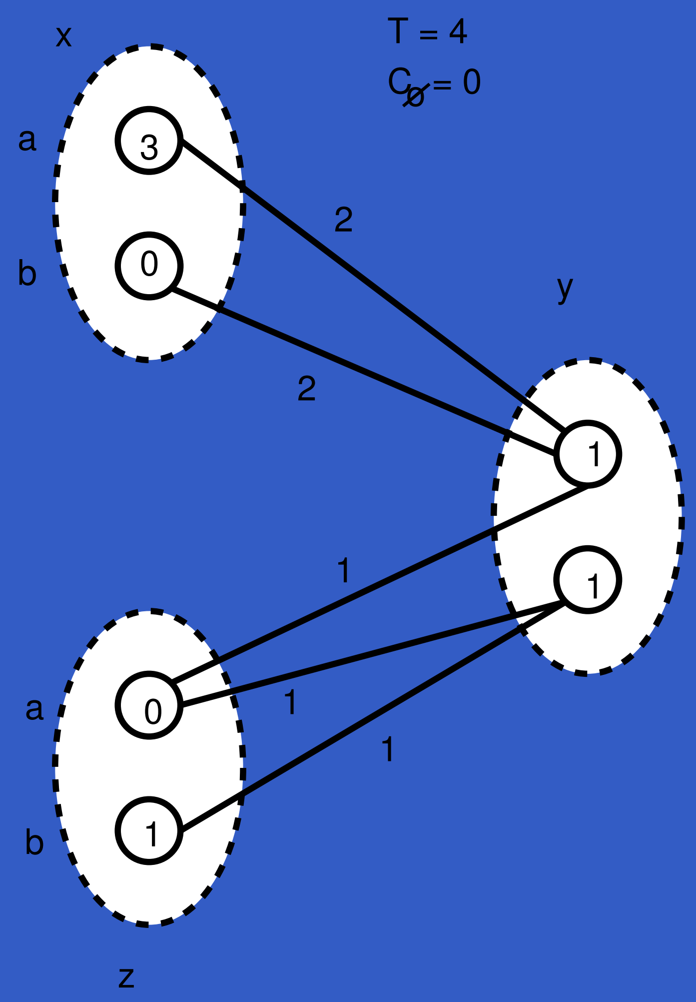
- NC: $O(nd)$
- AC: $O(n^2d^3)$ — AC > NC
- DAC: $O(ed^2)$ — DAC > NC
- FDAC: $O(end^3)$ — FDAC >AC, FDAC > DAC

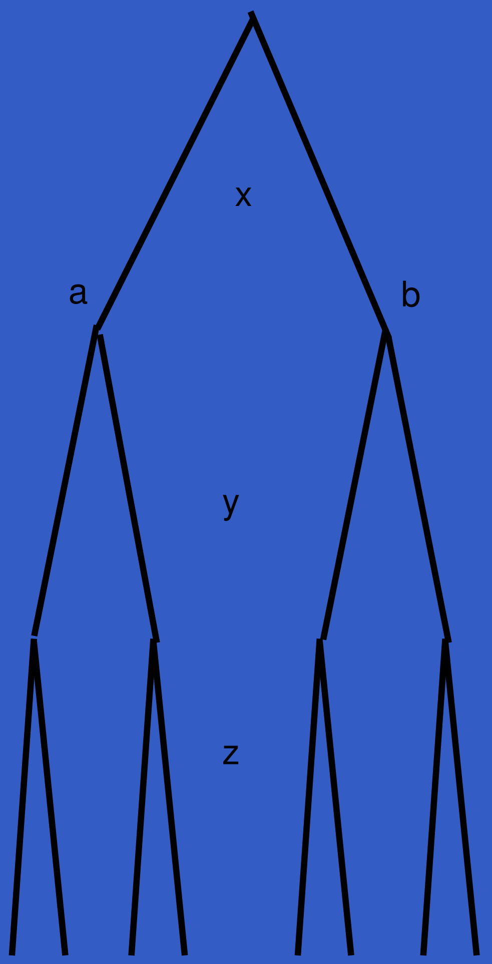
Integrating local consistencies in B & B

- at each node we have a VCSP with branching constraints.
- the ub gives the $\top$
- $c_\varnothing$ gives the lb

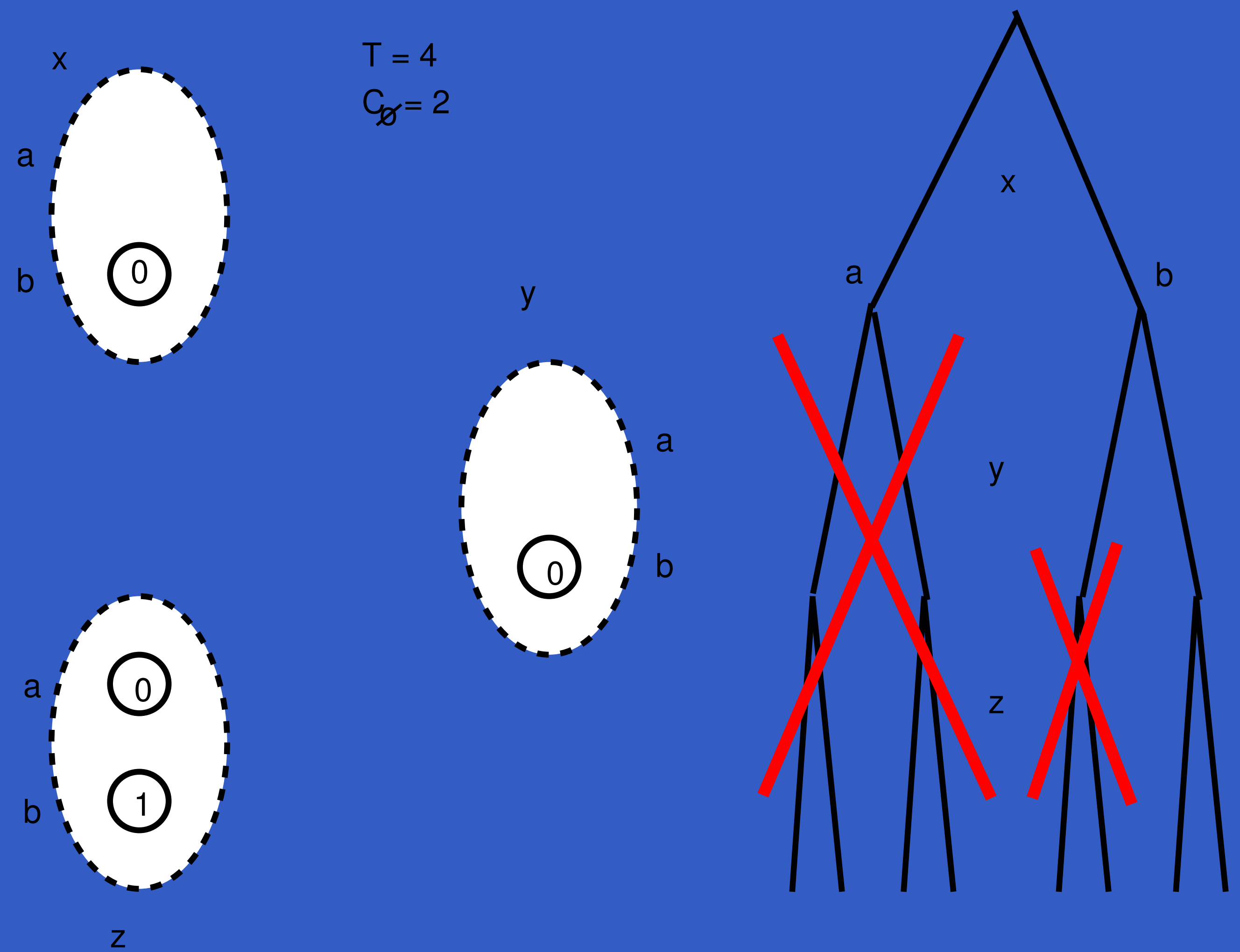
We can enforce NçAC, DAC, FDAC at every node and backtrack when wipe out ($\top$ and $c_\varnothing$ meet).

Example on AC





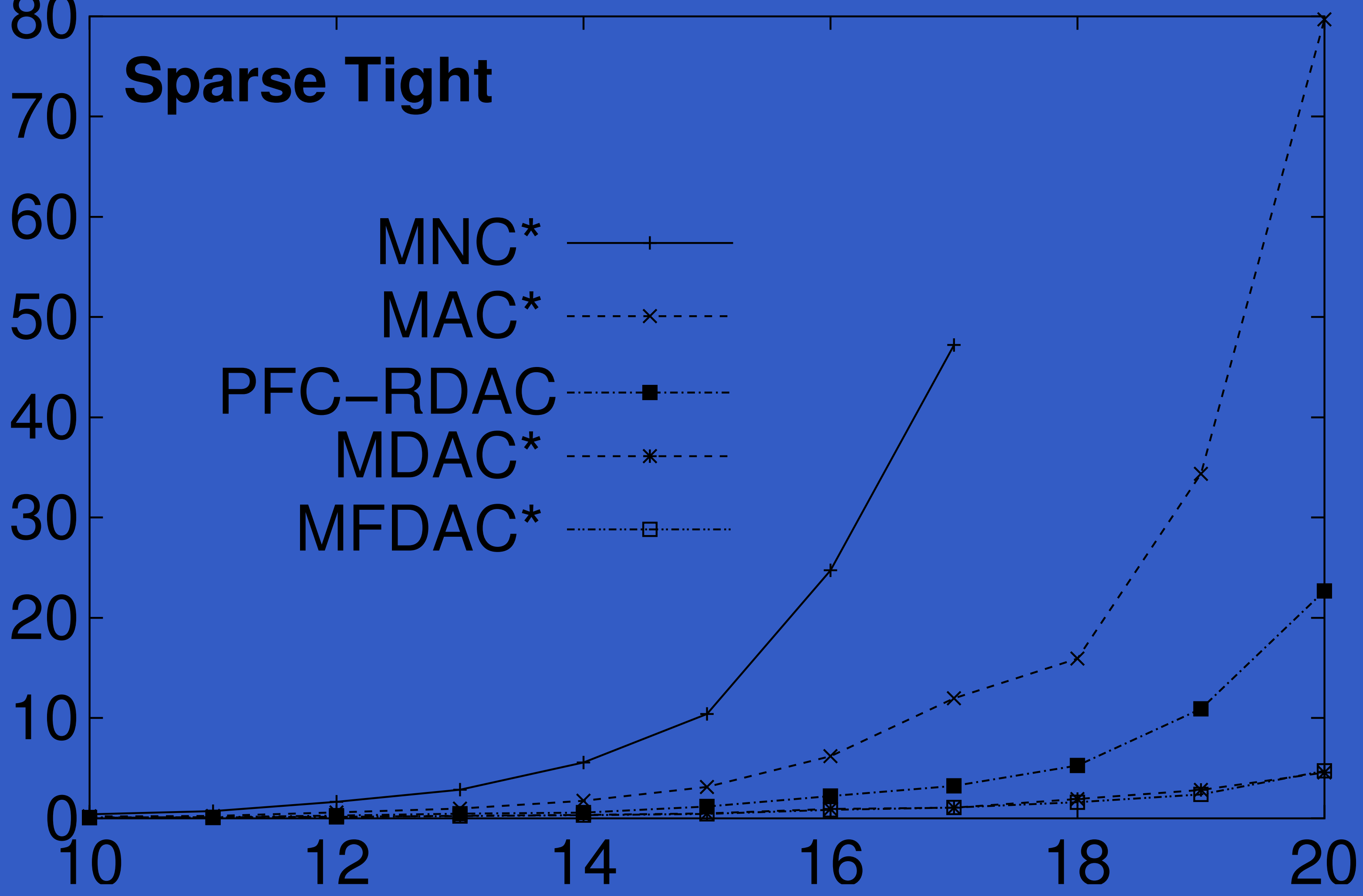
Example on AC



Practical comparison

- Using random overconstrained CSP
- Max-CSP: maximize the number of satisfied constraints
- Obvious WCSP translation (forbidden tuple have cost 1)
- Compare with previous algorithms (PFC-RDAC)

CPU-time on Sparse tight problems



Not always that good (simple problems).

Conclusion

- Local consistency extends simply to idempotent cases
- for non idempotent, we must compensate. Higher order (k-consistency) undefined yet.
- provides practically interesting lb.