An Abstraction Framework for Soft constraints, and its Relationship with Constraint Propagation

S. Bistarelli
University of Pisa, Italy

P. Codognet
University of Paris VI, France

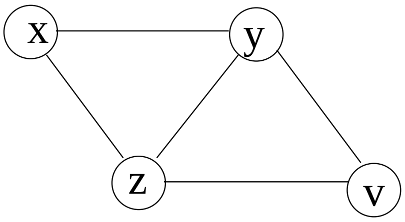
F. Rossi
University of Padova, Italy

Outline:

- soft constraints
- soft constraint propagation
- our abstraction approach
- properties of the abstraction
- relationship with constraint propagation

Classical constraints (CSPs)

- Variables over finite domains
- Constraints: restrictions over the values that some variables can take (tuples of values are either allowed or not)



All domains: r (red), b (blue), y (yellow)

All constraints: different

Solutions:

x	y	z	v
r	b	y	r
r	y	b	r
b	y	r	b
b	r	y	b
y	r	b	y
y	b	r	y

- Solution: assignment of all the variables which satisfies all constraints.

What are classical constraints useful for?

- Main features:
  - declarativeness
  - partial information
- Applications:
  - resource allocation
  - scheduling
  - vehicle routing
  - car sequencing
  - timetabling
  - graph coloring
  - VLSI design
  - vision
  - robotics
  - ...

However, in most real-life situations we need to express fuzziness, possibilities, preferences, probabilities, costs, ...

Examples of Soft Problems

- Over-constrained problems $\Rightarrow$ it does not make sense to just say that there is no solution
- Optimization problems
- Multi-criteria problems
- Problems with both preferences and hard statements (ex.: time-tabling with hard constraints about space and time, and soft constraints about teachers)
- Problems with uncertainties
- ...

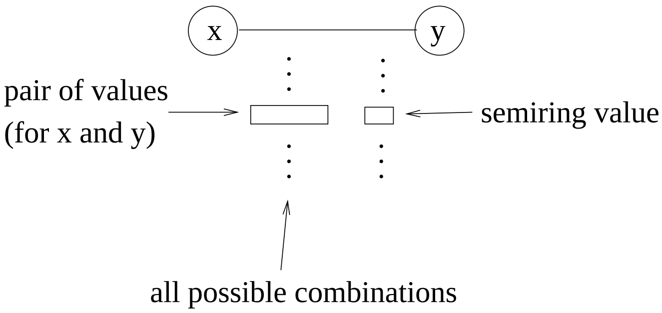
In general, human specifications are difficult to express faithfully via hard statements only.

Soft Constraint Satisfaction

- **Soft** constraints: each constraint has a level of importance
  - **hierarchical**: a total order of levels (HCLP of Borning, ICLP89)
  - **partial** order of levels (Chiu, Lee)
- **Prioritized** constraints: each tuple has a level of importance
- **Optimization** constraint solving: each constraint or tuple has a cost, to be minimized
- **Fuzzy** constraint solving: each tuple has a value between 0 and 1, to be maximized (Ruttkay94, Schiex92, Prade93,...)
- **Probabilistic** constraint solving: each constraint has a probability (of being in the real problem), to be maximized (Fargier 1993)
- **Partial** constraint solving: as classical, but maybe not all constraints are satisfied, thus metric space over all subsets of constraints (Frueder AIJ 1992)
- **Valued** CSPs: each constraint (or tuple) has a value taken from a totally ordered commutative monoid (Schiex IJCAI95)
- **Semiring-based** CSPs: similar to valued CSPs, but partially ordered (Bistarelli, Montanari, Rossi IJCAI95)

Semiring-based constraint solving

- From crisp constraints (either satisfied or not) to soft constraints: fuzzy, partial, probabilistic, soft, prioritized, optimization,...
- Each tuple of values of the involved variables satisfies a constraint with a certain level of preference, or cost, or degree, ...

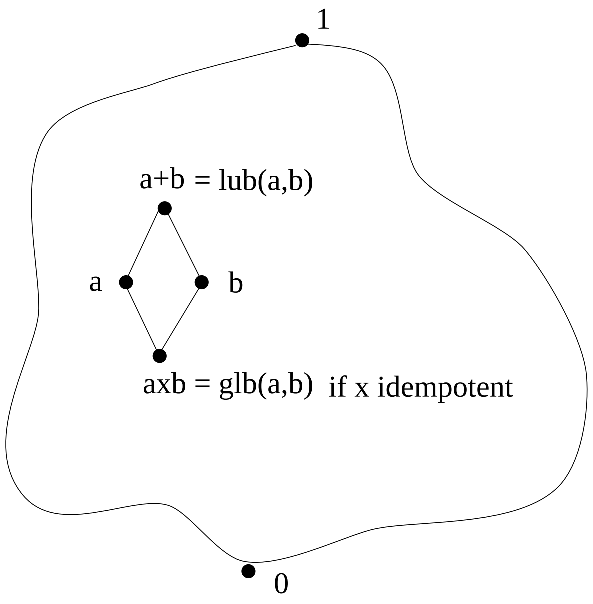


- By replacing the usual boolean algebra $\langle \{T, F\}, \vee, \wedge \rangle$ with any set, we get a semiring $\langle A, +, \times, 0, 1 \rangle$
- Additional properties we require:
  - $+$ idempotent, to get a partial order over A: $a \leq b$ (a is better than b) iff $a + b = b$; Then, 0 minimum;
  - $\times$ commutative, to consider sets of constraints instead of tuples;
  - $a + 1 = 1$. Then, 1 maximum, which implies $a \times b \leq a$.
- We have to use $+$ and $\times$ in place of $\vee$ and $\wedge$.

Properties of semiring-based CSPs

In a semiring $\langle A, +, \times, 0, 1\rangle$, we have:

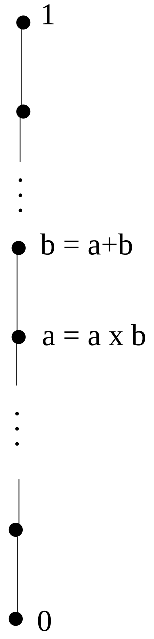
- $+$ and $\times$ monotone on $\leq$
- for all $a, b \in A$, $a \times b \leq a$
- for all $a \in A$, $\mathbf{0} \leq a \leq \mathbf{1}$
- $\langle A, \leq\rangle$ lattice, where
  - $+$ is the sup
  - if $\times$ idempotent, then $\times$ is the inf



- constraint: $\langle def, con\rangle$, $con \subseteq V$ (type), $def : D^k \rightarrow A$ (value), thus an element of A for each k-tuple of elements of D
- constraint problem: $\langle C, con\rangle$

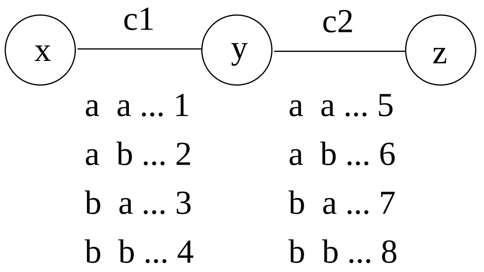
Total order

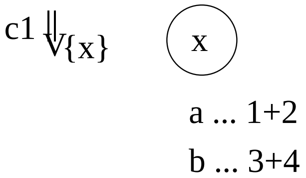
- when $\leq$ is a total order and $\times$ idempotent, we have that: if $a \leq b$, then $a + b = b$ and $a \times b = a$
- in other words: $a + b$ and $a \times b$ are either a or b

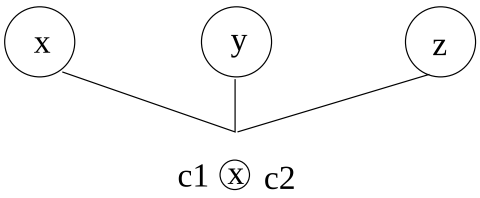


Combination and Projection

- **Projection**:
  $c \Downarrow_I = \langle def', I \cap con \rangle$, $def'(t') = \Sigma_{\{t \mid t \downarrow^{con}_{I \cap con} = t\}} def(t)$
- **Combination**:
  $c_1 \otimes c_2 = \langle def, con_1 \cup con_2 \rangle$, $def(t) = def_1(t \downarrow^{con}_{con_1}) \times def_2(t \downarrow^{con}_{con_2})$





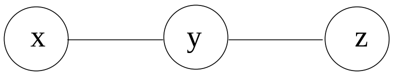


a	a	a	... 1x5
a	a	b	... 1x6
a	b	a	... 2x7
a	b	b	... 2x8
b	a	a	... 3x5
b	a	b	... 3x6
b	b	a	... 4x7
b	b	b	... 4x8

- $Sol(\langle C, con \rangle) = (\otimes C) \Downarrow_{con}$, thus an element of A for each con-tuple of elements of D

Examples

- CSPs: $\langle \{false, true\}, \vee, \wedge, false, true \rangle$:
  - *true* means the the tuple is allowed, *false* that is not allowed,
  - $\wedge$ combines tuples (the combination is allowed if all the combined tuples are so),
  - $\vee$ projects variables out (the shorter tuple is allowed if there is at least one extension which is so)
- Fuzzy CSPs: $\langle [0, 1], max, min, 0, 1 \rangle$:
  - tuples get values between 0 (worst value) and 1 (best value),
  - combination is achieved via minimization (the value of the combination is the min of the values of the combined tuples),
  - projection via maximization (the value of the shorter tuple is the max of the values of all extensions).



a a ... 0	a a ... 0.3
a b ... 1	a b ... 0.1
b a ... 0.5	b a ... 1
b b ... 0.7	b b ... 1

Solutions:

a a a ... min(0,0.3) = 0
a a b ... min(0,0.1) = 0
a b a ... min(1,1) = 1
a b b ... min(1,1) = 1
b a a ... min(0.5,0.3) = 0.3
b a b ... min(0.5,0.1) = 0.1
b b a ... min(0.7,1) = 0.7
b b b ... min(0.7,1) = 0.7

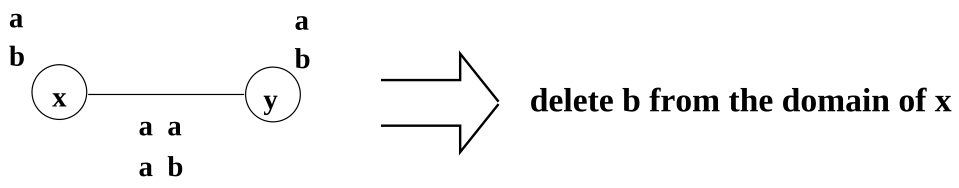
Best solutions:

a b a ... 1
a b b ... 1

How to solve soft problems?

By adapting and generalizing the techniques used for classical CSPs:

- backtracking search $\Rightarrow$ branch & bound
  (but also other techniques, like local search)
- constraint propagation:
  - for extensive constraints: it deletes elements from domains or constraints by considering a small subset of the constraints



  - for intensive contraints: it allows to derive new constraints

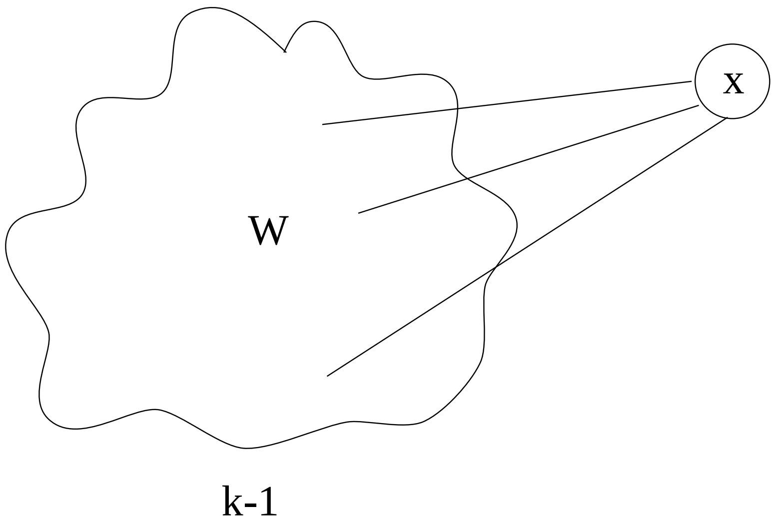
$$X = Y \wedge Y < Z \Rightarrow X < Z$$

$\Rightarrow$ propagation also for soft constraints.

Soft constraint propagation

- To eliminate some inconsistencies prior or during the search
- $\langle C, con\rangle$ is **k-consistent** if, for all subsets of k variables W and any other variable x:

$$\otimes\{c_i \mid c_i \in C \wedge con_i \subseteq W\}$$
$$=$$
$$(\otimes\{c_i \mid c_i \in C \wedge con_i \subseteq (W \cup \{x\})\}) \Downarrow_W$$

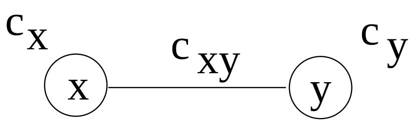


- Considering only the constraints in W is the same as considering all those in W, plus those connecting x to W, and then projecting over W

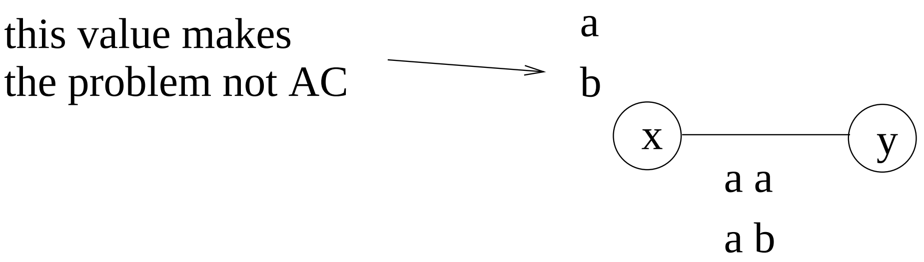
Semiring-based arc consistency (SAC)

- For arc-consistency (that is, k=2), k-consistency reduces to $W = \{y\}$, and

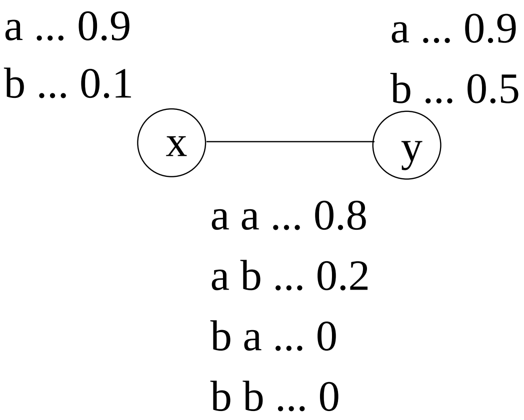
$$c_x = (\otimes\{c_y, c_{xy}, c_x\}) \Downarrow_x$$



- Over classical CSPs: any value in the domain of x can be extended to a value in the domain of y such that c_{xy} is satisfied;



- Over fuzzy CSPs: the level given to a value for x by c_x is the same as that given by $(\otimes\{c_y, c_{xy}, c_x\}) \Downarrow_x$

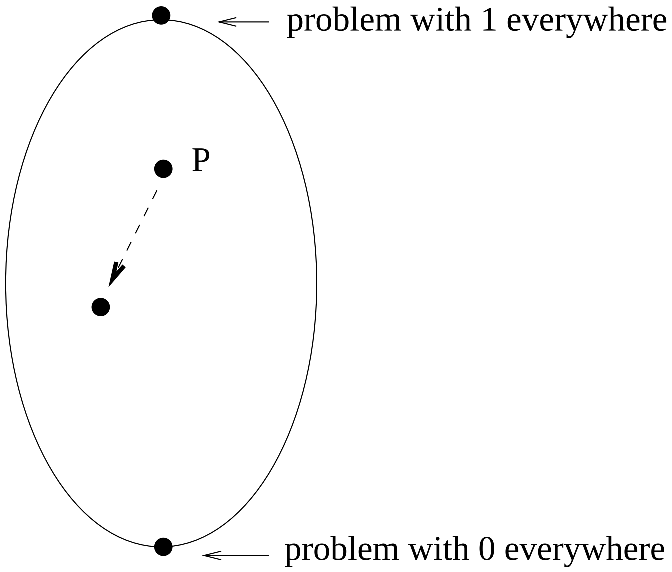


- To achieve SAC: for each x and y, change the definition of c_x to make it coincide with $(\otimes\{c_y, c_{xy}, c_x\}) \Downarrow_x$, and iterate until stability.

Properties of soft constraint propagation

1. $\times$ idempotent $\Rightarrow$ equivalence, termination, order-independence
2. given S, partial order $\sqsubseteq_S$ of problems with
   - same topology
   - same semiring S
   - $\sqsubseteq$ defined by extending $\leq_S$

   $\Rightarrow$ soft constraint propagation brings to lower problems



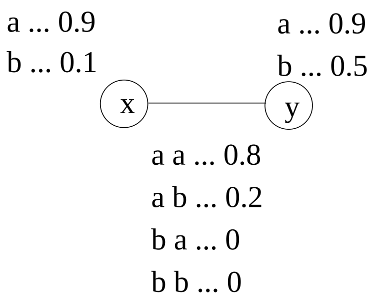
   - Some domain values will possibly be associated to smaller semiring elements
   - special case of CSPs: this means deleting these domain values (passing from *true* to *false*)

Examples

- CSPs: $\langle \{0,1\}, \vee, \wedge, 0, 1\rangle$:
  - $\wedge$ is idempotent, thus equivalence and order-independence
  - $\{0,1\}$ is a finite set, thus termination
- fuzzy CSPs: $\langle [0,1], max, min, 0, 1\rangle$:
  - min is idempotent, thus equivalence and order-independence
  - min is AD-closed for any finite subset AD of $[0,1]$, thustermination
- probabilistic CSPs: $\langle [0,1], max, \times, 0, 1\rangle$: $\times$ not idempotent and not closed on finite subsets of [0,1], thus no meaning to k-consistency algorithm
- optimization problems: $\langle \mathcal{R}^+, min, +, +\infty, 0\rangle$: $+$ not idempotent and not closed, thus no k-consistency

SAC on fuzzy CSPs

- Fuzzy CSPs: $\langle [0,1], max, min, 0, 1 \rangle$:
  - min is idempotent, thus equivalence and order-independence
  - min is AD-closed for any finite subset AD of $[0,1]$, thus termination



- Solutions: pairs, whose value is obtained via min. Ex.: $\langle a, a \rangle$ gets 0.8.
- This fuzzy CSP is not SAC:
  - c_x gives 0.9 to $x = a$
  - $(\otimes\{c_y, c_{xy}, c_x\}) \Downarrow_x$ gives it 0.8:
    * first combine c_x, c_{xy}, and c_y:

      a a ... min(0.9,0.8 ,0.9) = 0.8
      a b ... min(0.9,0.2,0.5) = 0.2
      b a ... min(0.1,0,0.9) = 0
      b b ... min(0.1,0,0.5) = 0

    * then project over x:

      a ... max(0.8,0.2) = 0.8
      b ... max (0,0) = 0

Abstracting soft constraints

- Soft constraints are more expressive and flexible than hard constraints, but also more difficult to process

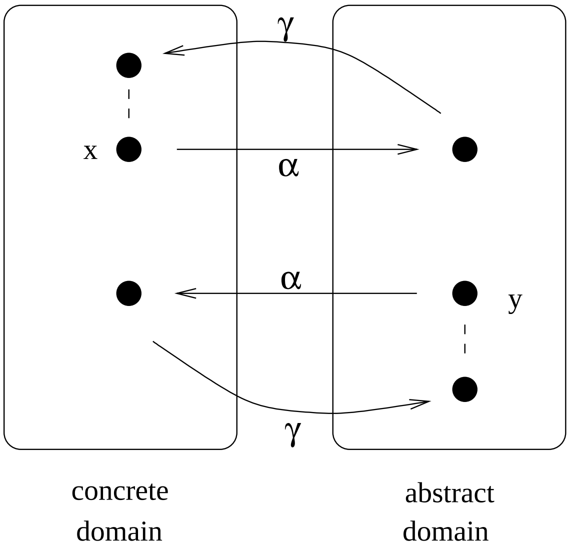
  $\Rightarrow$

  1. process an abstracted version
  2. derive useful information
  3. bring back to the original problem some of this information
  4. continue the solution process on the new problem.

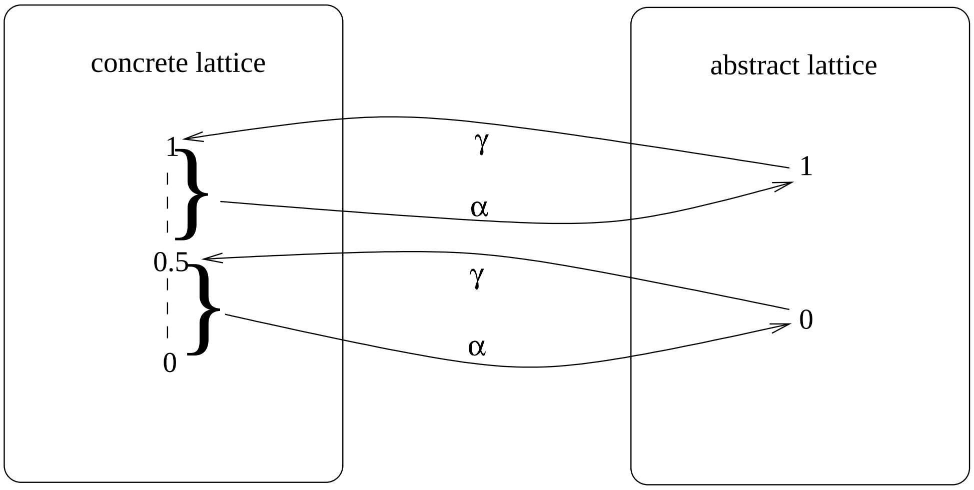
- Hope: that doing 1+2+3+4 is more efficient than 4 alone!

Abstraction tool

- **Galois insertion** between two partially ordered sets $\langle C, \sqsubseteq \rangle$ (the concrete domain) and $\langle A, \leq \rangle$ (the abstract domain):
  - abstraction function $\alpha : C \to A$
  - concretization function $\gamma : A \to C$
  - α and γ monotone
  - for each $x \in C, x \sqsubseteq (\gamma \circ \alpha)(x)$
  - for each $y \in A, (\alpha \circ \gamma)(y) \leq y$



Example of a Galois insertion



SCSP abstraction

- To abstract a soft constraint problem: just abstract its semiring!

  $$\Rightarrow$$

  we cannot change the topology of the problem (variables and constraint shape), nor the domains of the variables, but we can change the semiring set, and also the semiring operations.

- Example: given a problem over

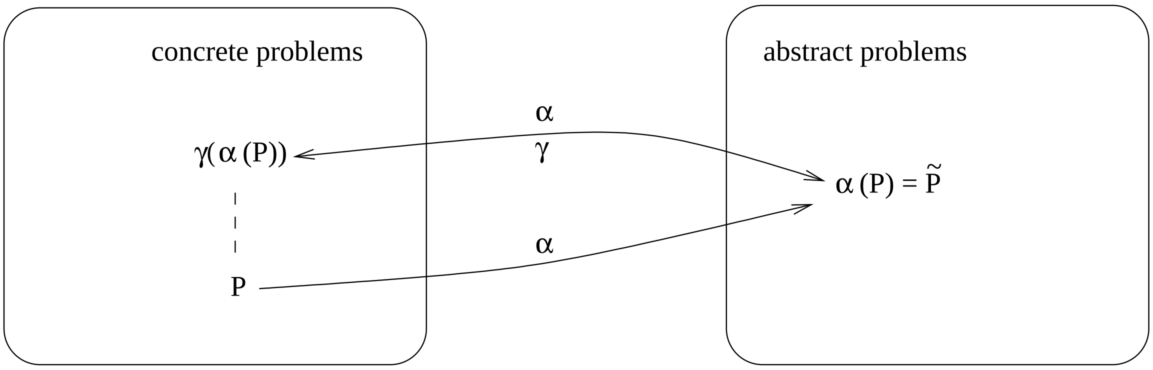
  $$\langle \mathcal{R}^+ \cup \{+\infty\}, \min, +, +\infty, 0\rangle \text{ (optimization)}$$

  we pass to a problem over $\langle [0,1], max, min, 0, 1\rangle$ (fuzzy)
  Possible reason: idempotence of $\times$.

- example: from fuzzy to classical. Possible reason: a more compact domain representation.

- Note: SCSP abstraction is transitive!

Properties of the abstraction – 1

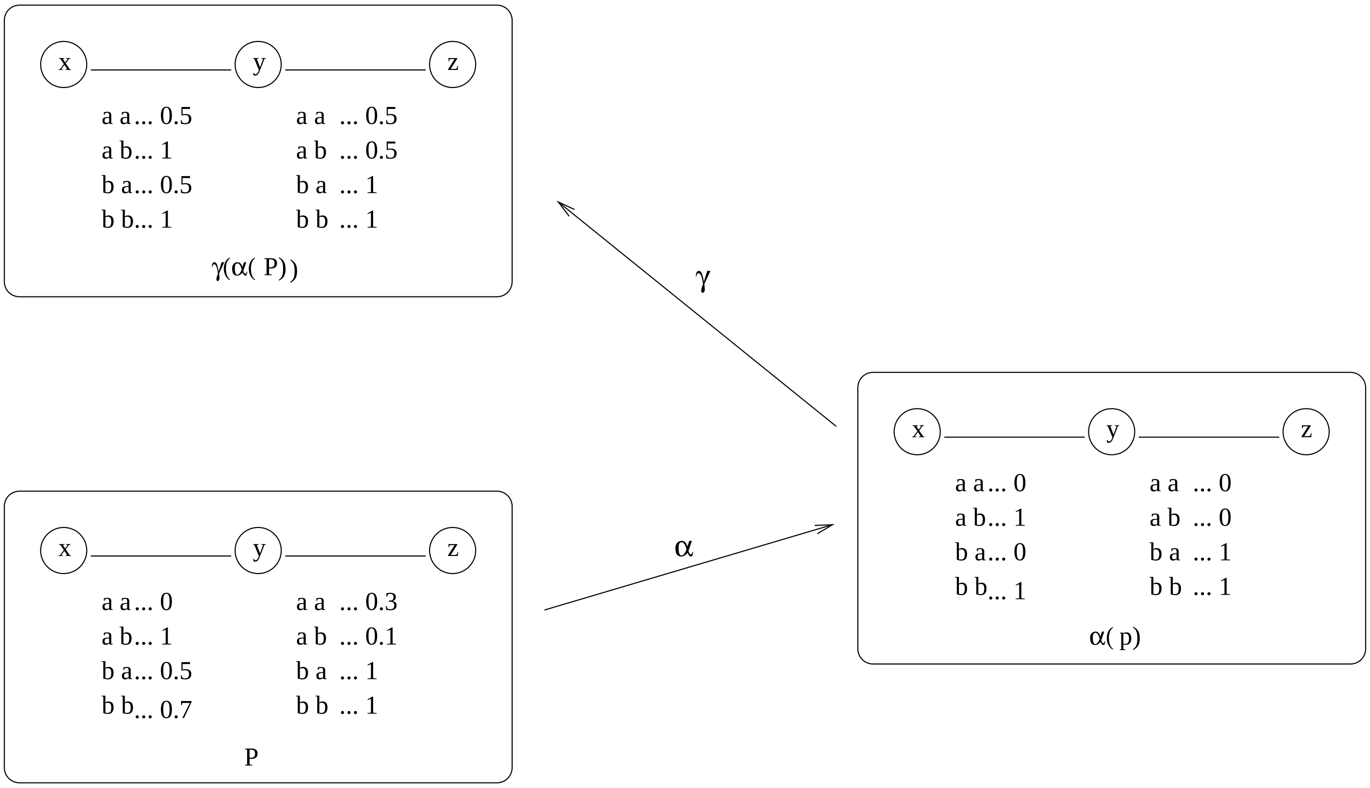


Theorem:

$$Sol(P) \sqsubseteq_S Sol(\gamma(\alpha(P))$$

⇒ we may loose some inconsistencies but we will not introduce new ones. If a solution has value 0 in the abstract side, then this is true also in the concrete side.

Example:



Order-preserving abstractions

- Two sets I_1 and I_2 of concrete elements
  $\Rightarrow$
  an abstraction is order-preserving if

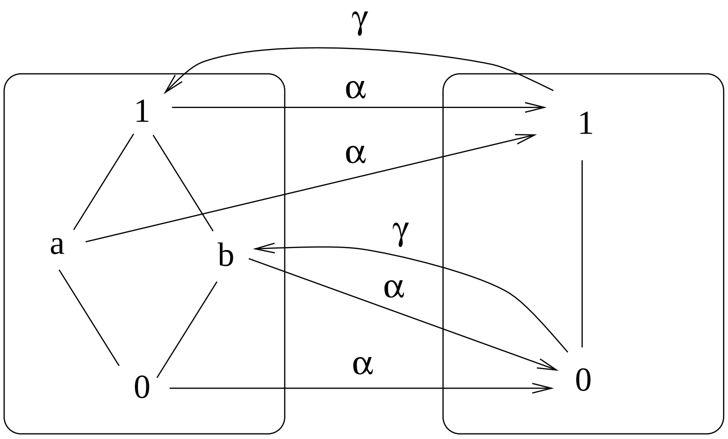
$$\tilde{\prod_{x \in I_1}} \alpha(x) \leq_{\tilde{S}} \tilde{\prod_{x \in I_2}} \alpha(x) \Rightarrow \prod_{x \in I_1} x \leq_S \prod_{x \in I_2} x$$

- If we first abstract and then combine, or we combine only, we get the same ordering among the resulting elements

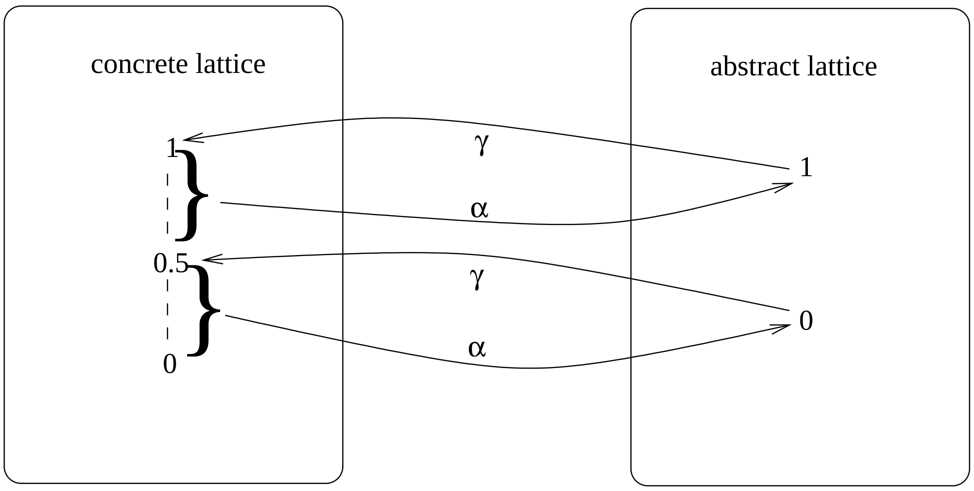
- $I_1 = \{x\}$ and $I_2 = \{y\}$
  $\Rightarrow \alpha(x) \leq_{\tilde{S}} \alpha(y)$ implies $x \leq_S y$
  $\Rightarrow$ if two abstract objects are ordered, then their concrete counterparts have to be ordered as well (and in the same way)

Examples

- A not-order-preserving abstraction:



- An order-preserving abstraction:



- A not-order-preserving abstraction:
  from $\langle N \cup \{+\infty\}, min, sum, 0, +\infty\rangle$ to $\langle N \cup \{+\infty\}, min, max, 0, +\infty\rangle$, with identity.

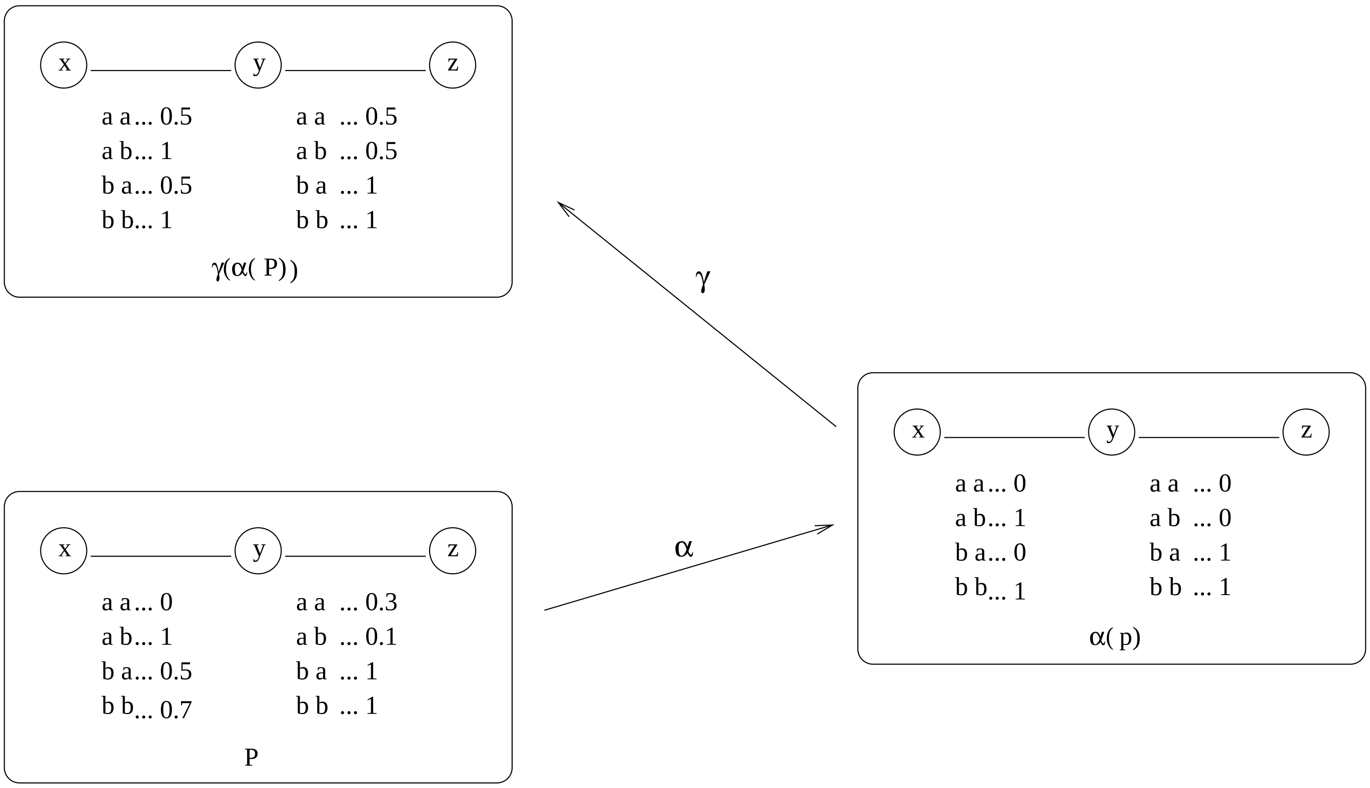
Properties of the abstraction – 2

Theorem: if the abstraction is order-preserving,

$$Opt(P) \subseteq Opt(\alpha(P))$$

$\Rightarrow$
to find an optimal solution of P, we may find all optimal solutions of $\alpha(P)$, and look among them on the concrete side.

Example:

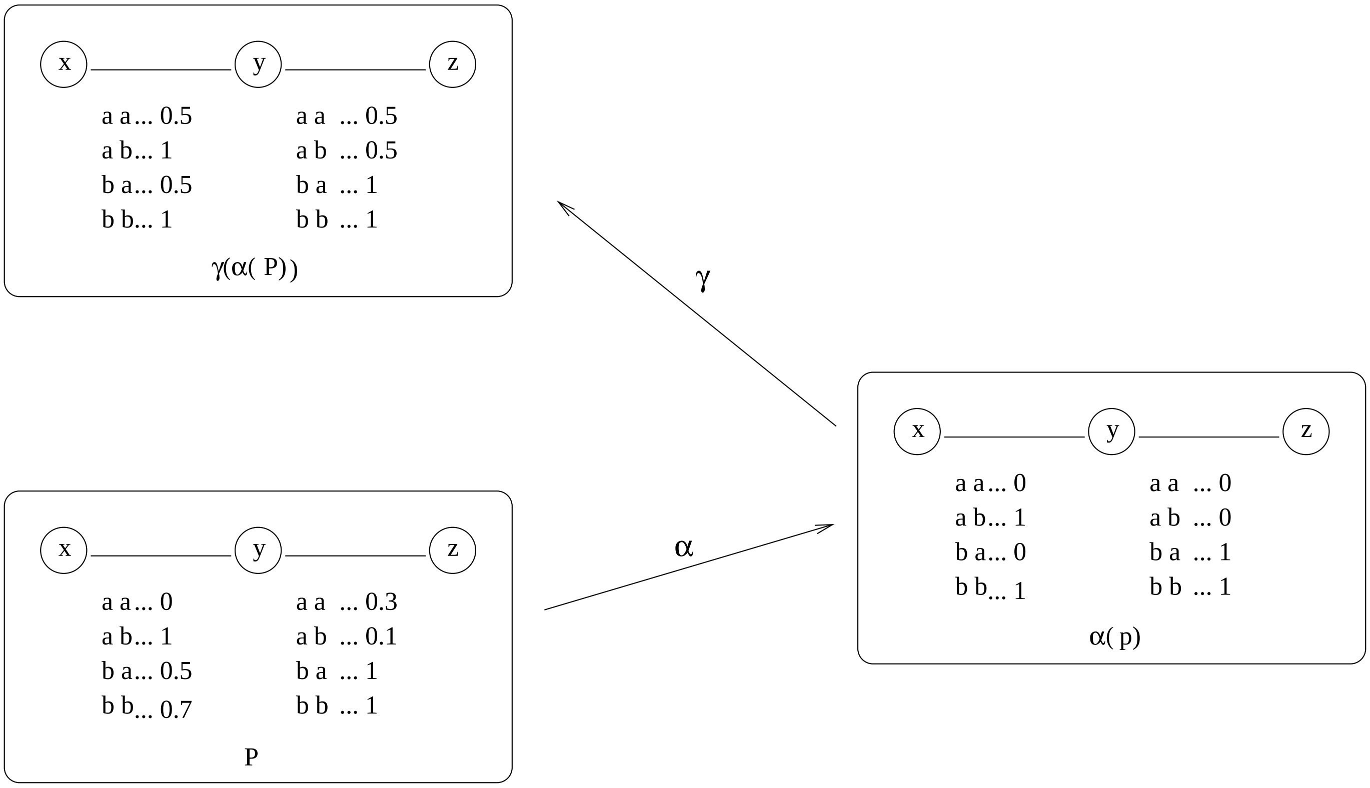


- optimal solutions in P: $\langle a, b, a\rangle$ and $\langle a, b, b\rangle$
- solutions in $\alpha(P)$: $\langle a, b, a\rangle$, $\langle a, b, b\rangle$, $\langle b, b, a\rangle$, $\langle b, b, b\rangle$

Properties of the abstraction – 3

Theorem: Given an optimal solution t for $\alpha(P)$, with value $\tilde{v}$, we can find upper and lower bounds for an optimal solution for P: there is an optimal solution in P with value between $\gamma(\tilde{v})$ and the value of t in P.

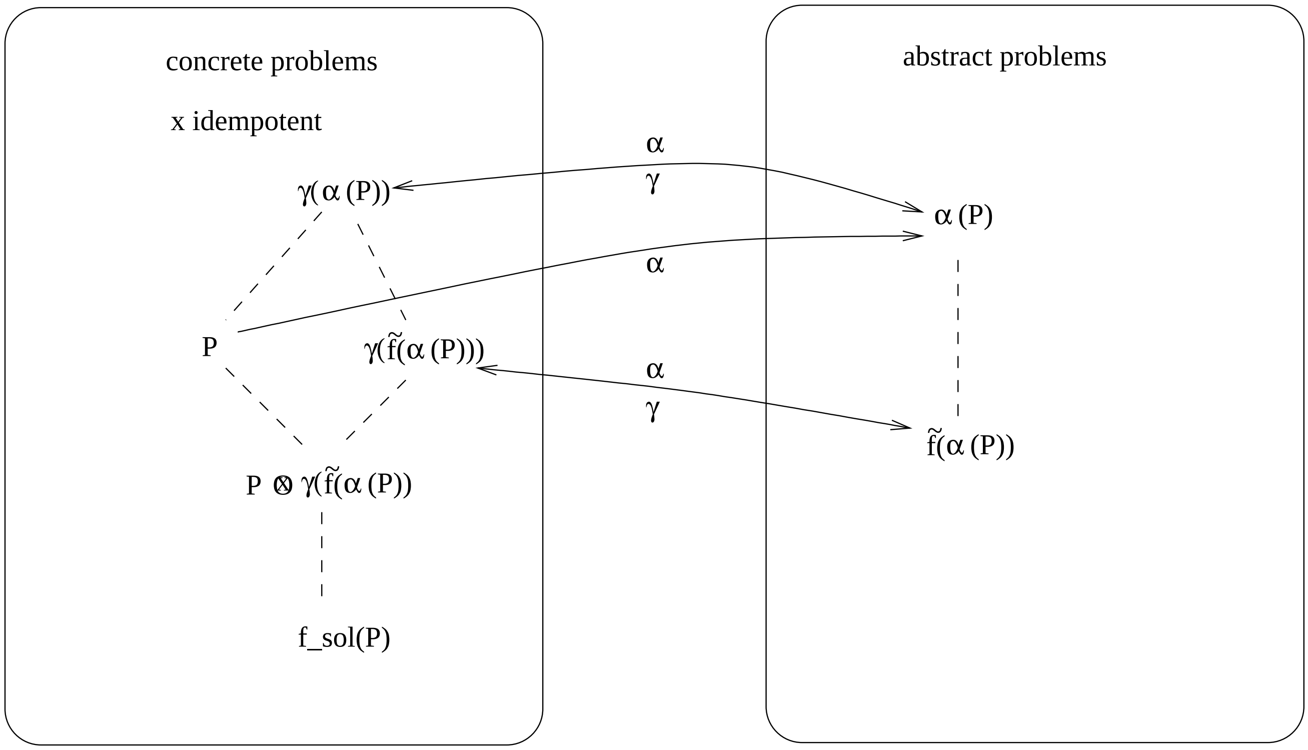
Example:



- Take any optimal solution of $\alpha(P)$, for example $\langle b, b, b \rangle$
- There exists an optimal solution of P with semiring value v between the value of $\langle b, b, b \rangle$ in P, which is 0.7, and $\gamma(1) = 1$
- In fact, there are optimal solutions with value 1 in P

Properties of the abstraction – 4

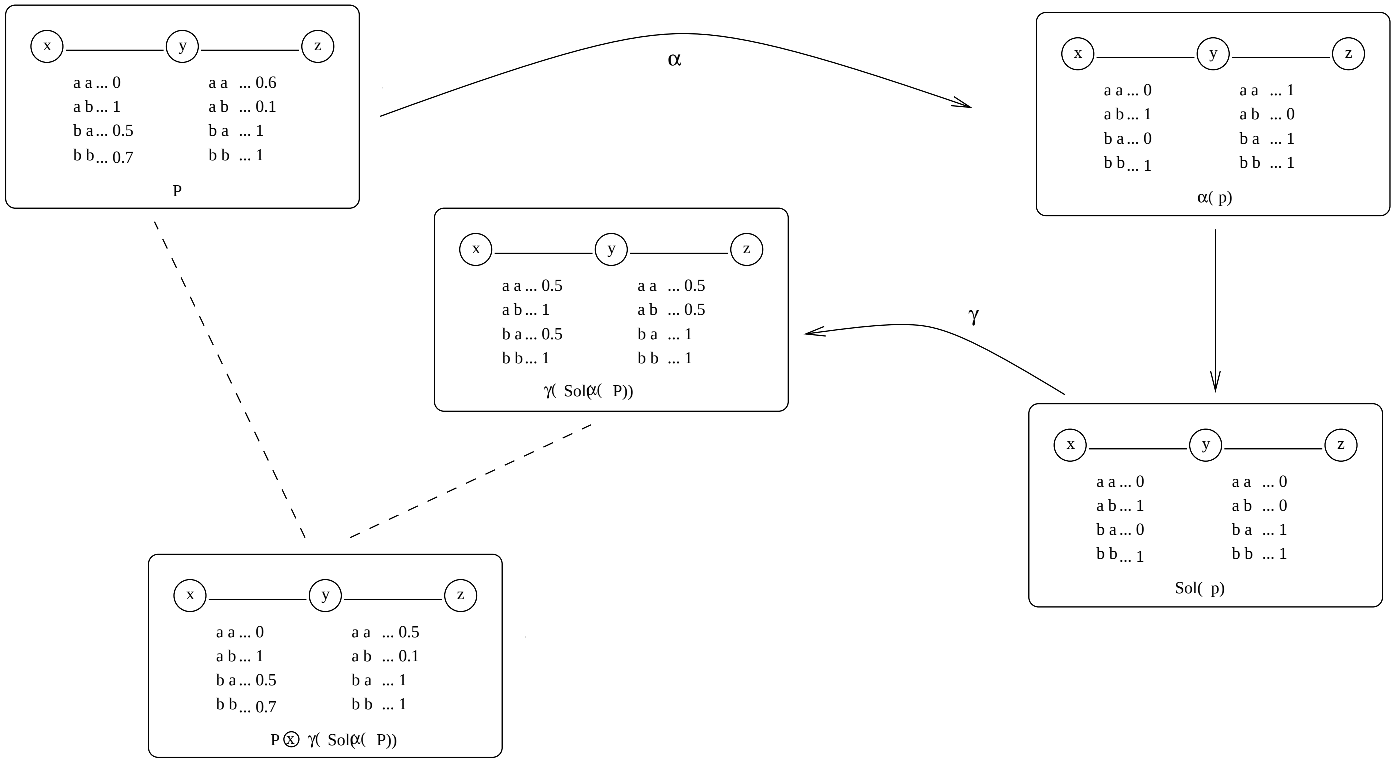
Suppose $\times$ idempotent.



1. from P to $\alpha(P)$
2. constraint propagation over $\alpha(P)$, obtaining $f(\alpha(P))$
3. we concretize: $\gamma(\tilde{f}(\alpha(P)))$
4. we combine with the original problem: $P \otimes \gamma(\tilde{f}(\alpha(P)))$

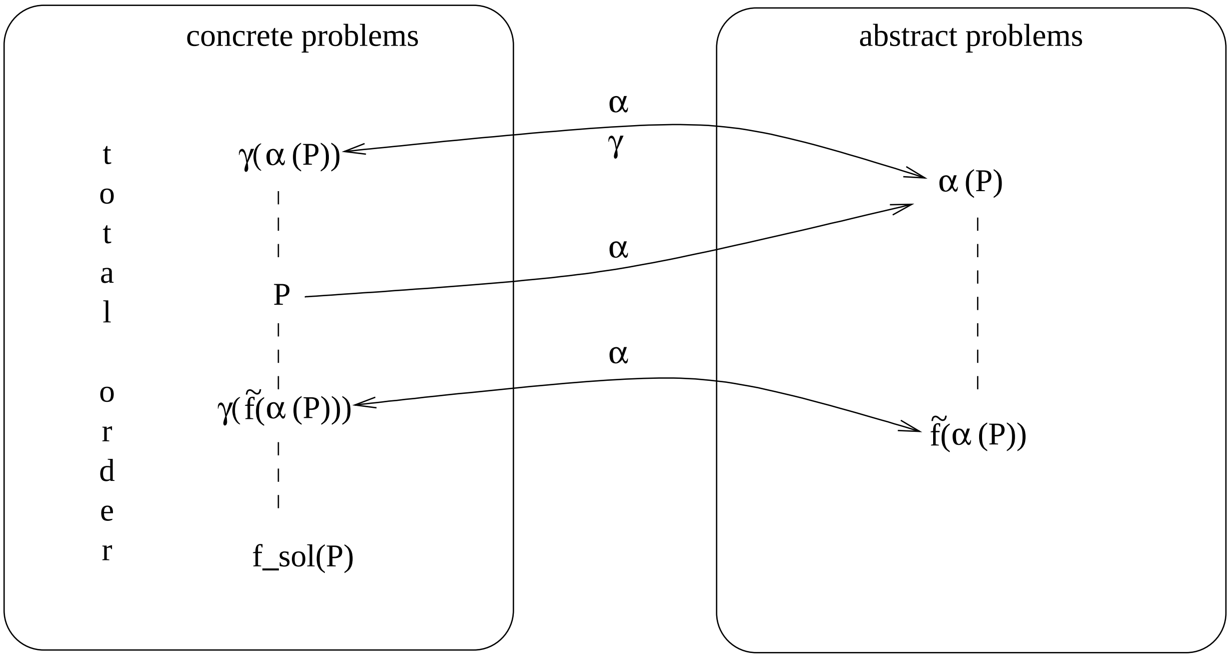
$\Rightarrow$ we have obtained an equivalent problem which is closer to the solution (a branch & bound algorithm explores a smaller search space).

Example


α
γ
x y z
a a ... 0 a a ... 0.6
a b ... 1 a b ... 0.1
b a ... 0.5 b a ... 1
b b ... 0.7 b b ... 1
P
a a ... 0 a a ... 1
a b ... 1 a b ... 0
b a ... 0 b a ... 1
b b ... 1 b b ... 1
α(p)
a a ... 0.5 a a ... 0.5
a b ... 1 a b ... 0.5
b a ... 0.5 b a ... 1
b b ... 1 b b ... 1
γ(Sol(α(P))
a a ... 0 a a ... 0
a b ... 1 a b ... 0
b a ... 0 b a ... 1
b b ... 1 b b ... 1
Sol(p)
a a ... 0 a a ... 0.5
a b ... 1 a b ... 0.1
b a ... 0.5 b a ... 1
b b ... 0.7 b b ... 1
P ⊗ γ(Sol(α(P))

Properties of the abstraction – 5

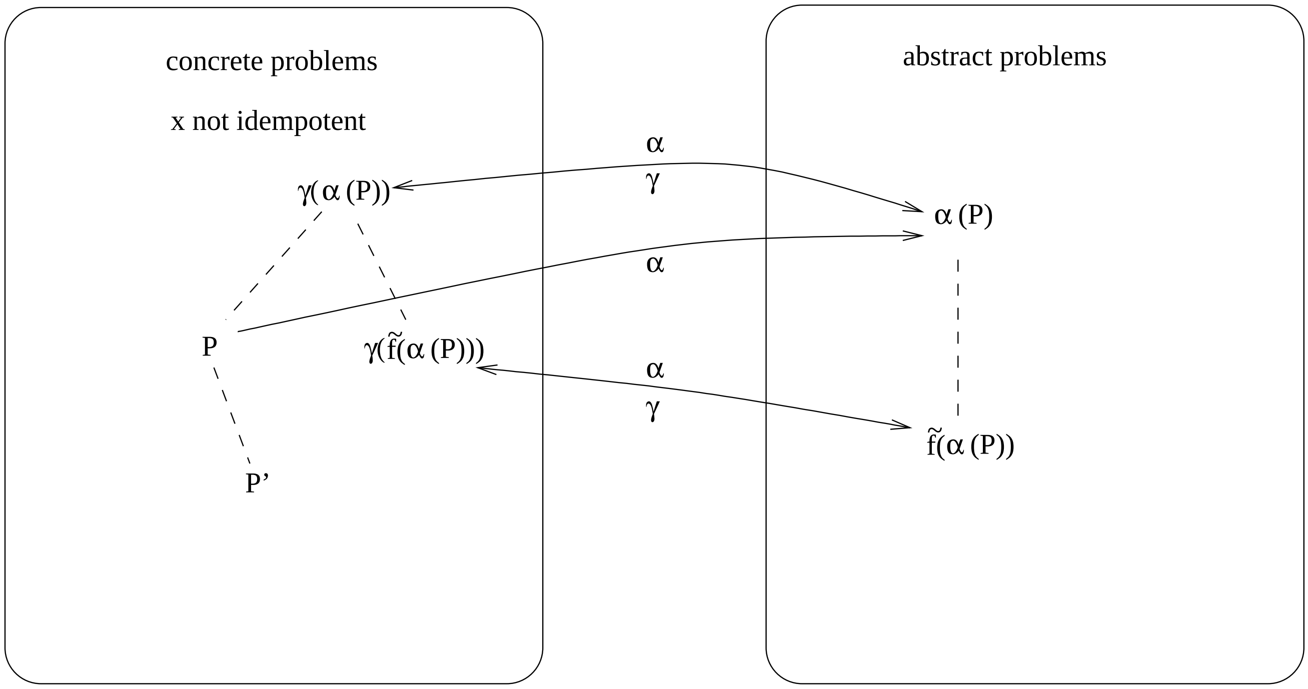
When $\leq$ is a total order:



$\Rightarrow$ we pass from P to $\gamma(\tilde{f}(\alpha(P)))$.

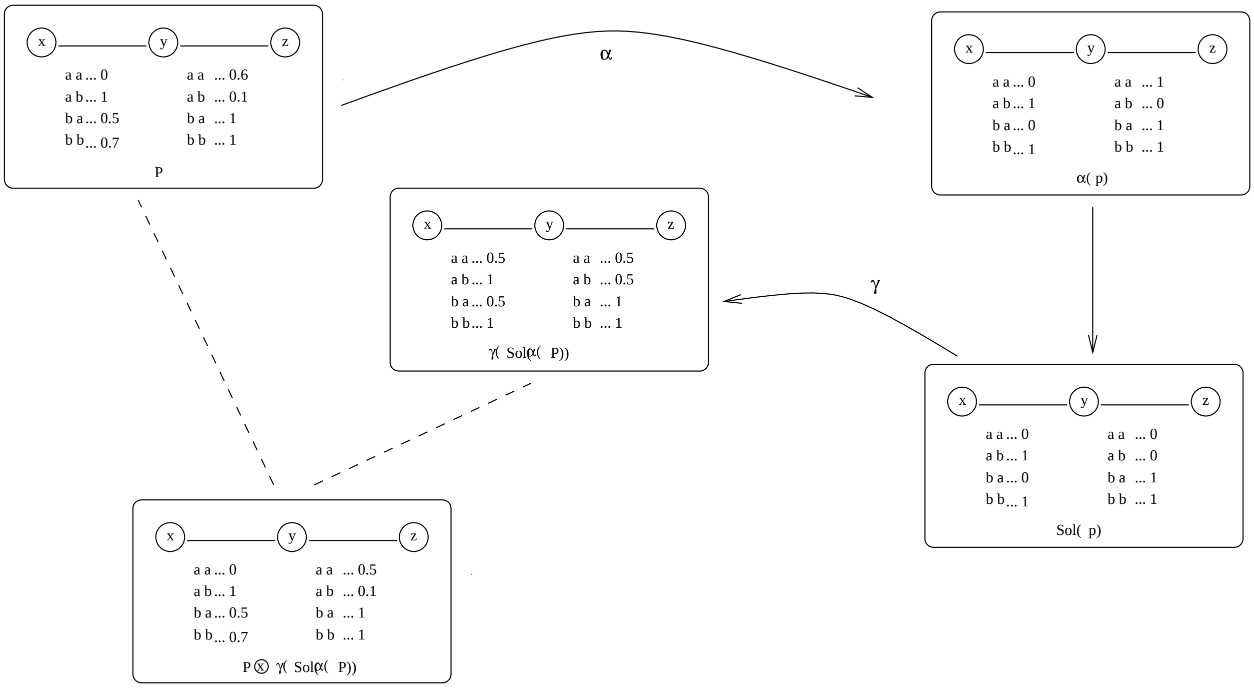
Properties of the abstraction – 6

When $\times$ is not idempotent:

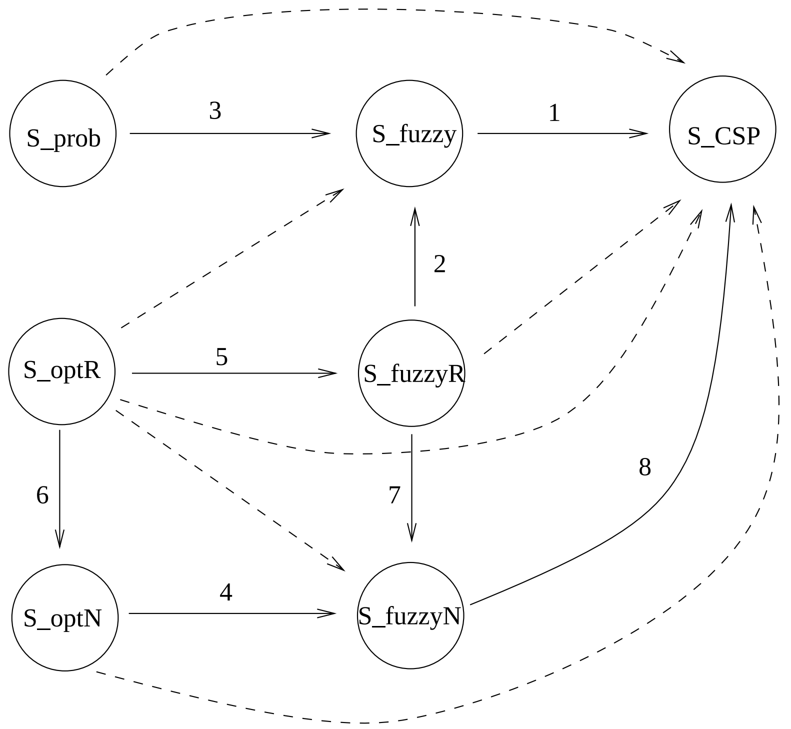


$\Rightarrow$ we pass from P to P', which is the same as P, except for those tuples with value 0 in $\gamma(\tilde{f}(\alpha(P)))$, that are given value 0 also in P' (we can inherit only the inconsistencies).

Example


x y z
a a ... 0
a b ... 1
b a ... 0.5
b b ... 0.7
a a ... 0.6
a b ... 0.1
b a ... 1
b b ... 1
P
α
x y z
a a ... 0
a b ... 1
b a ... 0
b b ... 1
a a ... 1
a b ... 0
b a ... 1
b b ... 1
α(p)
x y z
a a ... 0.5
a b ... 1
b a ... 0.5
b b ... 1
a a ... 0.5
a b ... 0.5
b a ... 1
b b ... 1
γ(Sol(α(P))
γ
x y z
a a ... 0
a b ... 1
b a ... 0
b b ... 1
a a ... 0
a b ... 0
b a ... 1
b b ... 1
Sol(p)
x y z
a a ... 0
a b ... 1
b a ... 0.5
b b ... 0.7
a a ... 0.5
a b ... 0.1
b a ... 1
b b ... 1
P ⊗ γ(Sol(α(P))

Some abstractions between interesting semirings



Relationship with constraint propagation

- Any order-preserving abstraction $\langle \alpha, \gamma \rangle$ between S and $\bar{S}$
- $\times_S$ idempotent
- Any propagation rule r in $\bar{S}$

$\Rightarrow \gamma(r(\alpha(P))) \otimes P$ is a propagation rule for P

- It could be useful when S does not have any, or any efficient, propagation algorithms
- We can use to the propagation algorithms of $\bar{S}$ to perform propagation also over S

The TI property

- Theorem-increasing (TI) property: what is a theorem in the concrete domain, remains a theorem in the abstract domain
- Desirable property in a general theory of abstraction (Giunchiglia-Walsh, AIJ 1992)
- In our framework, a theorem is an SCSP which has a solution with value greater than or equal to k, where $k \geq 0$
- Our version of the TI property (which holds):
  If P which has a solution with value $v \geq k$, then $\alpha(P)$ has a solution with value $v' \geq \alpha(k)$.

Conclusions

- Soft CSPs: a general framework to model many real-life situations (probabilities, optimization, fuzziness, partiality, uncertainties, ...)
- Abstraction to reduce the complexity of handling soft constraints

Future Work

- Define some α/γ pairs for specific pairs of semirings, and find their properties.
- Implement the abstraction framework, and assess its convenience in terms of time to find an optimal (or acceptable) solution
- Combine this abstraction framework with other notions of abstractions (for domains, or for graphs, ...)