DM841

Constraint Programming

Notions of Local Consistency

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Constraint Programming

The **domain** of a variable x, denoted $D(x)$, is a finite set of elements that can be assigned to x.

A **constraint** C on X is a subset of the Cartesian product of the domains of the variables in X, i.e., $C \subseteq D(x_1) \times \cdots \times D(x_k)$. A tuple $(d_1, \ldots, d_k) \in C$ is called a solution to C.

Equivalently, we say that a solution $(d_1, ..., d_k) \in C$ is an assignment of the value d_i to the variable x_i for all $1 \leq i \leq k$, and that this assignment satisfies C.

If $C = \emptyset$, we say that it is inconsistent.

Constraint Programming

Constraint Satisfaction Problem (CSP)

A CSP is a finite set of variables X with domain extension $\mathcal{D} = D(x_1) \times \cdots \times D(x_n)$, together with a finite set of constraints $\mathcal{C}$, each on a subset of X. A **solution** to a CSP is an assignment of a value $d \in D(x)$ to each $x \in X$, such that all constraints are satisfied simultaneously.

Constraint Optimization Problem (COP)

A COP is a CSP $\mathcal{P}$ defined on the variables $x_1, \ldots, x_n$, together with an objective function $f : D(x_1) \times \cdots \times D(x_n) \to Q$ that assigns a value to each assignment of values to the variables. An **optimal solution** to a minimization (maximization) COP is a solution d to $\mathcal{P}$ that minimizes (maximizes) the value of $f(d)$.

Constraint Propagation

Definition (Domain consistency)

For a CSP, a constraint C on the variables $x_1, \ldots, x_k$ is called domain consistent if for each variable x_i and each value $v_i \in D(x_i)$ $(i = 1, \ldots, k)$, there exists a value $v_j \in D(x_j)$ for all $j \neq i$ such that $(v_1, \ldots, v_i, \ldots, v_k) \in C$.

Loose definition

Domain filtering is the removal of values from variable domains that are not consistent with an individual constraint.

Constraint propagation is the repeated application of all domain filtering of individual constraints until no domanin reduction is possible anymore.

Task:

- determine whether the CSP/COP is consistent (has a solution):
- find one solution
- find all solutions
- find one optimal solution
- find all optimal solutions

Solving CSPs

- Systematic search:
  - choose a variable x_i that is not yet assigned
  - create a choice point, i.e. a set of mutually exclusive & exhaustive choices, e.g. $x_i = v$ vs $x_i \neq v$
  - try the first & backtrack to try the other if this fails
- Constraint propagation:
  - add $x_i = v$ or $x \neq v$ to the set of constraints
  - re-establish "local" consistency on each constraint
    $\rightsquigarrow$ remove values from the domains of future variables that can no longer be used because of this choice
  - fail if any future variable has no values left

Representing a Problem

- a CSP $\mathcal{P} = \langle X, \mathcal{D}, \mathcal{C} \rangle$ represents a problem P, if every solution of $\mathcal{P}$ corresponds to a solution of P and every solution of P can be derived from at least one solution of $\mathcal{P}$
- More than one solution of $\mathcal{P}$ can represent the same solution of P or viceversa, if symmetries are present
- The variables and values of $\mathcal{P}$ represent entities in P
- The constraints of $\mathcal{P}$ ensure the correspondence between solutions
- we must make sure that any solution to $\mathcal{P}$ yields exactly one solution to P, and that any solution to P corresponds to a solution to $\mathcal{P}$ or is symmetrically equivalent to such a solution, and that if $\mathcal{P}$ has no solutions, this is because P itself has no solutions.
- The aim is to find a model $\mathcal{P}$ that can be solved as quickly as possible
(Note that shortest run-time might not mean least search!)

Interactions with Search Strategy

Whether a model is better than another can depend on the search algorithm and search heuristics

- Let's assume that the search algorithm is fixed
  although different level of consistency can also play a role
- Let's also assume that choice points are always $x_i = v$ vs $x_i \neq v$
- Variable (and value) order still interact with the model a lot
- Is variable & value ordering part of modelling?

In practice it is.
but it depends on the modeling language used

Outline

Goals

- Establish formalism around constraint handling
- Learning general constraint propagation algorithms

Reasoning with Constraints

Constraint Propagation, related notions:

- constraint relaxation
- filtering algorithms
- narrowing algorithms
- constraint inference
- simplification algorithms
- label inference
- local consistency enforcing
- rules iteration
- propagation loop
- proof rules

Local Consistency define properties that the constraint problem must satisfy after constraint propagation

Rules Iteration defines properties on the process of propagation itself, that is, kind and order of operations of reduction applied to the problem

Notation and Terminology

Finite domains $\rightsquigarrow$ w.l.g. $D \subseteq \mathbb{Z}$

Constraint C: relation on a (ordered) *subsequence* of variables

- $X(C) = (x_{i_1}, \ldots, x_{i_{|X(C)|}})$ is the scheme or scope of C
- $|X(C)|$ is the arity of C (unary/binary/non-binary)
- $C \subseteq \mathbb{Z}^{|X(C)|}$ containing combinations of valid values (or tuples) $\tau \in \mathbb{Z}^{|X(C)|}$
- constraint check: testing whether a τ satisfies C
- $\mathcal{C}$: a t-tuple of constraints $\mathcal{C} = (C_1, \ldots, C_t)$
- expression
  - extensional: specifies satisfying tuples (aka `table` or `regular` via DFA). eg. $c(x_1, x_2) = \{(2,2), (2,3), (3,2), (3,3)\}$
  - intensional: specifies the characteristic function. eg. $\texttt{alldiff}(x_1, x_2, x_3)$

CSP

Input:

- **Variables** $X = (x_1, \ldots, x_n)$
- **Domain Expression** $\mathcal{D} = \{x_1 \in D(x_1), \ldots, x_n \in D(x_n)\}$
- $\mathcal{C}$ finite set of constraints each on a subsequence Y of X.
  $C \in \mathcal{C}$ on $Y = (y_1, \ldots, y_k)$ is $C \subseteq D(y_1) \times \ldots \times D(y_k)$

a constraint satisfaction problem (CSP) is

$$\mathcal{P} = \langle X,\ \mathcal{D},\ \mathcal{C} \rangle$$

$(v_1, \ldots, v_n) \in D(x_1) \times \ldots \times D(x_n)$ is a solution of $\mathcal{P}$
if for each constraint $C_i \in \mathcal{C}$ on $x_{i_1} \ldots, x_{i_m}$ it is

$$(v_{i_1}, \ldots, v_{i_m}) \in C_i$$

CSP normalized: iff two different constraints do not involve exactly the same vars
CSP binary iff for all $C_i \in \mathcal{C}, |X(C)| = 2$

Notation and Terminology

Given a tuple τ on a sequence Y of variables and $W \subseteq Y$,

- $\tau[W]$ is the restriction of τ to variables in W (ordered accordingly)
- $\tau[x_i]$ is the value of x_i in τ
- if $X(C) = X(C')$ and $C \subseteq C'$ then for all $\tau \in C$ the reordering of τ according to $X(C')$ satisfies C'.

Example

$C(x_1, x_2, x_3): \quad x_1 + x_2 = x_3$
$C'(x_1, x_2, x_3): \quad x_1 + x_2 \leq x_3$

$C \subseteq C'$

Notation and Terminology

- Given $Y \subseteq X(C)$, $\pi_Y(C)$ denotes the projection of C on Y. It contains tuples on Y that can be extended to a tuple on $X(C)$ satisfying C.
- given $X(C_1) = X(C_2)$, the intersection $C_1 \cap C_2$ contains the tuples τ that satisfy both C_1 and C_2
- join of $\{C_1 \dots C_k\}$ is the relation with scheme $\cup_{i=1}^{k} X(C_i)$ that contains tuples such that $\tau[X(C_i)] \in C_i$ for all $1 \leq i \leq k$.

Example

$$\mathcal{P} = \langle X = (x_1, x_2, x_3, x_4), \mathcal{D} = \{D(x_i) = \{1..5\}, \forall i\},$$
$$\mathcal{C} = \{C_1 \equiv \texttt{alldiff}(x_1, x_2, x_3), C_2 \equiv x_1 \leq x_2 \leq x_3, C_3 \equiv x_4 \geq 2x_2\}\rangle$$

$\pi_{\{x_1, x_2\}}(C_1) \equiv (x_1 \neq x_2)$
$C_1 \cap C_2 \equiv (x_1 < x_2 < x_3)$
join $\{C_1, \dots, C_3\} \equiv (x_1 < x_2 < x_3 \wedge x_4 \geq 2x_2)$

Notation and Terminology

Given $\mathcal{P} = \langle X, \mathcal{D}, \mathcal{C} \rangle$ the instantiation I is a tuple on $Y = (x_1, \ldots, x_k) \subseteq X$: $((x_1, v_1), \ldots, (x_k, v_k))$

- I on Y is valid iff $\forall x_i \in Y$, $I[x_i] \in D(x_i)$
- I on Y is locally consistent iff it is valid and for all $C \in \mathcal{C}$ with $X(C) \subseteq Y$, $I[X(C)]$ satisfies C (some constraints may have $X(C) \not\subseteq Y$)
- a solution to $\mathcal{P}$ is an instantiation I on $X(\mathcal{C})$ which is locally consistent
- I on Y is globally consistent if it can be extended to a solution, i.e., there exists $s \in sol(\mathcal{P})$ with $I = s[Y]$

Example

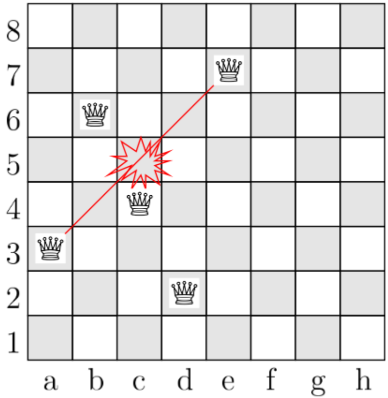
$$\mathcal{P} = \langle X = (x_1, x_2, x_3, x_4), \mathcal{D} = \{D(x_i) = \{1..5\}, \forall i\},$$
$$\mathcal{C} = \{C_1 \equiv \texttt{alldiff}(x_1, x_2, x_3), C_2 \equiv x_1 \leq x_2 \leq x_3, C_3 \equiv x_4 \geq 2x_2\}\rangle$$

$\pi_{\{x_1, x_2\}}(C_1) \equiv (x_1 \neq x_2)$
$I_1 = ((x_1, 1), (x_2, 2), (x_4, 7))$ is not valid
$I_2 = ((x_1, 1), (x_2, 1), (x_4, 3))$ is local consistent since C_3 only one with $X(C_3) \subseteq Y$ and $I_2[X(C_3)]$ satisfies C_3
I_2 is not global consistent: $sol(\mathcal{P}) = \{(1, 2, 3, 4), (1, 2, 3, 5)\}$

Notation and Terminology

- An instantiation I on $\mathcal{P}$ is globally consistent if it can be extended to a solution of $\mathcal{P}$, globally inconsistent otherwise.
- A globally inconsistent instantiation is also called a (standard) nogood.
  (a partial instantiation that does not lead to a solution.)
- Remark: A locally inconsistent instantiation is a nogood. The reverse is not necessarily true
- A nogood is a globally inconsistent instantiation, which can be locally consistent or inconsistent.

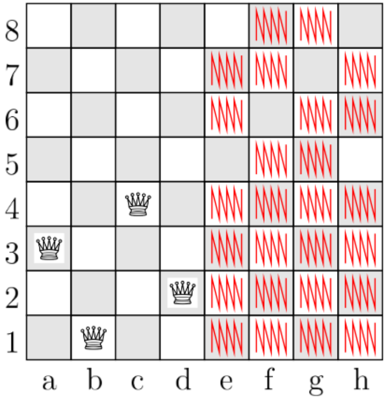
Example



$\{(x_a,3),(x_b,6),(x_c,4),(x_d,2),(x_e,7)\}$ is locally inconsistent

☞ this is a nogood.

Example



$\{(x_a,3),(x_b,1),(x_c,4),(x_d,2)\}$ is globally inconsistent

☞ this is a nogood.

Notation and Terminology

CSP solved by extending partial instantiations to global consistent ones and backtracking at local inconsitencies $\rightsquigarrow$ is NP-complete!

Idea: make the problem more explicit (tighter)

$\mathcal{P}' \preceq \mathcal{P}$ iff $X_{\mathcal{P}'} = X_{\mathcal{P}}$ and any instantiation I on $Y \subseteq X_{\mathcal{P}}$ locally inconsistent for $\mathcal{P}$ is locally inconsistent for $\mathcal{P}'$.

Example

$$\begin{aligned}\mathcal{P} = &\langle X = (x_1, x_2, x_3), \mathcal{D} = \{D(x_i) = [1..4], \forall i\},\\ &\quad \mathcal{C} = \{C_1 \equiv x_1 < x_2, C_2 \equiv x_2 < x_3,\\ &\quad C_3 \equiv \{(111), (123), (222), (333), (234)\}\}\rangle\\ \mathcal{P}' = &\langle X, \mathcal{D}, \mathcal{C}'\rangle, \mathcal{C}' = \{C_1, C_2, C_3' \equiv \{(123)\}\}\rangle\end{aligned}$$

$\mathcal{P}' \preceq \mathcal{P}$: All locally inconsitent instantiations on $Y \subseteq X_{\mathcal{P}}$ for $\mathcal{P}$ are locally inconsistent for $\mathcal{P}'$.
Indeed $X_{\mathcal{P}'} = X_{\mathcal{P}}, \quad \mathcal{D}_{\mathcal{P}'} = \mathcal{D}$ and $C_1 = C_1', C_2 = C_2', X(C_3) = X(C_3'), C_3' \subset C_3$.
However not all solutions are preserved!

Example

$$\mathcal{P} = \langle X = (x_1, x_2, x_3), \mathcal{D} = \{D(x_i) = [1..4], \forall i\},$$
$$\mathcal{C} = \{C_1 \equiv x_1 < x_2, C_2 \equiv x_2 < x_3, C_3 \equiv \{(111), (123), (222), (333)\}\}\rangle$$

$$\mathcal{P}' = \langle X, \mathcal{D}, \mathcal{C}' \rangle, \mathcal{C}' = \{C_1, C_2, C_3' \equiv \{(123), (231), (312)\}\}$$

For any tuple τ on $X(C)$ that does not satisfy C there exists a constraint C' in $\mathcal{C}'$ with $X(C') \subseteq X(C)$ such that $\tau[X(C')] \notin C'$ (τ local inconsistent). Hence $\mathcal{P}' \preceq \mathcal{P}$. But also $\mathcal{P} \preceq \mathcal{P}'$. They are no-good equivalent.

$\rightsquigarrow$ $\preceq$ does not define an order, just a preorder (antisymmetry does not hold.)

So with this definition:
– we may miss solutions
– we may not even get an easier problem

Constraint Propagation

Constraint Propagation transforms a problem $\mathcal{P}$ by tightening $\mathcal{D}$, by tightening constraints from $\mathcal{C}$ or by adding new constraints to $\mathcal{C}$. It does not remove redundant constraints which is a modeling task.

$\mathcal{P}$' is a tightening of $\mathcal{P}$ (and by implication $\mathcal{P}' \preceq \mathcal{P}$) if
$X_{\mathcal{P}'} = X_{\mathcal{P}}, \quad \mathcal{D}_{\mathcal{P}'} \subseteq \mathcal{D}, \quad \forall C \in \mathcal{C}, \exists C' \in \mathcal{C}', X(C') = X(C)$ and $C' \subseteq C$.

Note that in the previous example $\mathcal{P}'$ is not a tightening of $\mathcal{P}$: $C_3' \not\subseteq$ of any $C \in \mathcal{C}$, neither viceversa. Tightening defines a non-strict weak order (total preorder).

Example

$$\begin{aligned}
\mathcal{P} =& \langle X = (x_1, x_2, x_3), \mathcal{D} = \{D(x_i) = [1..4], \forall i\}, \\
& \mathcal{C} = \{C_1 \equiv x_1 < x_2, C_2 \equiv x_2 < x_3, \\
& C_3 \equiv \{(111), (123), (222), (333), (234)\}\}\rangle \\
\mathcal{P}' =& \langle X, \mathcal{D}, \mathcal{C}' \rangle, \mathcal{C}' = \{C_1, C_2, C_3' \equiv \{(123)\}\}\rangle
\end{aligned}$$

$\mathcal{P}' \preceq \mathcal{P}$: $X_{\mathcal{P}'} = X_{\mathcal{P}}, \quad \mathcal{D}_{\mathcal{P}'} = \mathcal{D}$ and $C_1 = C_1', C_2 = C_2', X(C_3) = X(C_3'), C_3' \subset C_3$.

Notation and Terminology

$\mathcal{S}_{\mathcal{P}}$ is the space of all tightening for $\mathcal{P}$

We are interested in the tightenings that preserve the set of solutions ($\mathrm{sol}(\mathcal{P}') = \mathrm{sol}(\mathcal{P})$) whose space is denoted $\mathcal{S}^{\mathrm{sol}}_{\mathcal{P}}$ and among them the smallest ($\mathcal{P}^* \preceq \mathcal{P}''$ for all $\mathcal{P}'' \in \mathcal{S}^{\mathrm{sol}}_{\mathcal{P}}$.)

$\mathcal{P}^* \in \mathcal{S}^{\mathrm{sol}}_{\mathcal{P}}$ is globally consistent if any instantiation I on $Y \subseteq X$ which is locally consistent in $\mathcal{P}^*$ can be extended to a solution of $\mathcal{P}$.

Computing $\mathcal{P}^*$ is exponential in time and space $\leadsto$ search a close $\mathcal{P}$ in polynomial time and space $\leadsto$ constraint propagation

- Define a property Φ that states necessary conditions on instantiations for solutions. Φ is called local consistency.
- Reduction rules: sufficient conditions to rule out values (or instantiations) that will not be part of a solution (defined through a consistency property Φ)
  Rules iteration: set of reduction rules for each constraint that tighten the problem

Constraint Propagation

In general, we reach a $\mathcal{P}$' that is Φ consistent by constraint propagation:

- tighten $\mathcal{D}$
- tighten $\mathcal{C}$, ex: $x_1 + x_2 \leq x_3 \rightsquigarrow x_1 + x_2 = x_3$
- add C to $\mathcal{C}$

Focus on domain-based tightenings

Domain-based tightenings

The space $\mathcal{S}_{\mathcal{P}}$ of domain-based tightenings of $\mathcal{P}$ is the set of problems $\mathcal{P}' = \langle X', \mathcal{D}', \mathcal{C}' \rangle$ such that $X_{\mathcal{P}'} = X_{\mathcal{P}}, \quad \mathcal{D}_{\mathcal{P}'} \subseteq \mathcal{D}, \quad \mathcal{C}' = \mathcal{C}$

As before the task is:
Finding a tightening $\mathcal{P}^*$ in $\mathcal{S}_{\mathcal{P}}^{\mathrm{sol}} \subseteq \mathcal{S}_{\mathcal{P}}$ (the set that contains all problems that preserve the solutions of $\mathcal{P}$) such that:
forall $x_i \in X_{\mathcal{P}}$, $D_{\mathcal{P}*}(x_i)$ contains only values that belong to a solution itself, i.e., $D_{\mathcal{P}*}(x_i) = \pi_{\{x_i\}}(\mathrm{sol}(\mathcal{P}))$

- Reduction rules:

$$D(x_i) \leftarrow D(x_i) \cap \{v_i \mid D(x_1) \times D(x_j - 1) \times \{v_i\} \times \ldots D(x_j + 1) \times \ldots D(x_k) \cap C \neq \emptyset\}$$

  (the rule is parameterised by a variable x_i and a constraint C)

- Rules iteration (for all i)

It is clearly NP-hard since it corresponds to solving $\mathcal{P}$ itself.
$\leadsto$ hence polynomial reduction rules to approximate $\mathcal{P}^*$

Apply rules iteration for each constraint. Domain-based reduction rules are also called propagators.

Example

$C = (|x_1 - x_2| = k)$
Propagator: $D(x_1) \leftarrow D(x_1) \cap [\min_D(x_2) - k .. \max_D(x_2) + k]$

Rather than defining rules we define Φ: e.g., unary, arc, path, k-consistency

Domain-based local consistency

Domain-based local consistency property Φ specifies a necessary condition on values to belong to solutions. We restrict to those stable under union.

A domain-based property Φ is stable under union iff for any Φ-consistent problem $\mathcal{P}_1 = \langle X, \mathcal{D}_1, \mathcal{C} \rangle$ and $\mathcal{P}_2 = \langle X, \mathcal{D}_2, \mathcal{C} \rangle$ the problem $\mathcal{P}' = \langle X, \mathcal{D}_1 \cup \mathcal{D}_2, \mathcal{C} \rangle$ is Φ-consistent.

Example

Φ for each constraint C and variable $x_i \in X(C)$, at least half of the values in $D(x_i)$ belong to a valid tuple satisfying C.

$$\mathcal{P}_1 = \langle X = (x_1, x_2), \mathcal{D} = \{D_1(x_1) = \{1, 2\}, D_1(x_2) = \{2\}\}, C \equiv \{x_1 = x_2\} \rangle$$
$$\mathcal{P}_2 = \langle X = (x_1, x_2), \mathcal{D} = \{D_2(x_1) = \{2, 3\}, D_2(x_2) = \{2\}\}, C \equiv \{x_1 = x_2\} \rangle$$

Both are Φ consistent but they are not stable under union.

Domain-based tightenings

Note: Not all Φ-consistent tightenings preserve the solutions
We search for the Φ-closure $\Phi(\mathcal{P})$ (the union of all Φ-consistent $\mathcal{P}' \in \mathcal{S}_{\mathcal{P}}$)

If Φ is stable under union, then $\Phi(\mathcal{P})$ is the unique domain-based Φ-consistent tightening problem that contains all others.

$\mathrm{sol}(\phi(\mathcal{P})) = \mathrm{sol}(\mathcal{P})$

Example

$$\mathcal{P} = \langle X = (x_1, x_2, x_3, x_4), \mathcal{D} = \{D(x_i) = \{1, 2\}, \forall i\}, \\ \mathcal{C} = \{C_1 \equiv x_1 \leq x_2, C_2 \equiv x_2 \leq x_3, C_3 \equiv x_1 \neq x_3\}\rangle$$

Φ all values for all variables can be extended consistently to a second variable

$$\mathcal{P}' = \langle X = (x_1, x_2, x_3, x_4), \mathcal{D} = \{D(x_1) = 1, D(x_2) = 1, D(x_3) = 2, \forall i\}, \\ \mathcal{C} = \{C_1 \equiv x_1 \leq x_2, C_2 \equiv x_2 \leq x_3, C_3 \equiv x_1 \neq x_3\}\rangle$$

$\mathcal{P}$' is consistent but it does not contain $(1, 2, 2)$ which is in $\mathrm{sol}(\mathcal{P})$
$\Phi(\mathcal{P}) : \langle X, \mathcal{D}_\Phi, \mathcal{C}\rangle$ with $D_\Phi(x_1) = 1, D_\Phi(x_2) = \{1, 2\}, D_\Phi(x_3) = 2$

Definition

A set is closed under an operation if performance of that operation on members of the set always produces a member of the same set.

A set is said to be closed under a collection of operations if it is closed under each of the operations individually.

Domain-based tightenings

Proposition (Fixed Point): If a domain based consistency property Φ is stable under union, then for any $\mathcal{P}$, the $\mathcal{P}$' with $\mathcal{D}_{\mathcal{P}'}$ obtained by iteratively removing values that do not satisfy Φ until no such value exists is the Φ-closure of $\mathcal{P}$.

Contrary to $\mathcal{P}^*$, $\Phi(\mathcal{P})$ can be computed by a greedy algorithm:

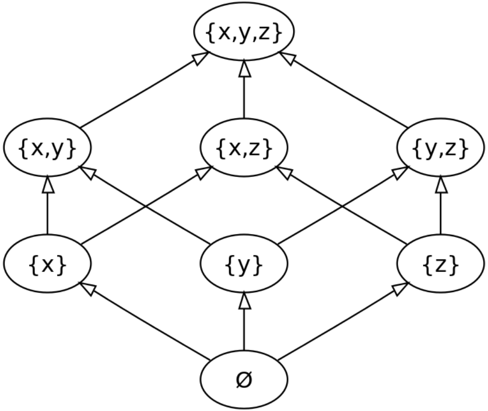
Corollary If a domain-based consistency property Φ is polynomial to check, finding $\Phi(\mathcal{P})$ is polynomial as well.

enforcing Φ consistency $\equiv$ finding closure $\Phi(\mathcal{P})$

Orders

Domain-based tightenings define a partial order (poset) because isomorphic to inclusion $\subseteq$, which is a partial order

(For a, b, elements of a poset P, if $a \leq b$ or $b \leq a$, then a and b are comparable. Otherwise they are incomparable)



Orders

Possible to define a partial order also on the local consistency property:

Definition

- Φ_1 is at least as strong as another Φ_2 if for any $\mathcal{P}$: $\Phi_1(\mathcal{P}) \leq \Phi_2(\mathcal{P})$:
ie, $X_{\Phi_1(\mathcal{P})} = X_{\Phi_2(\mathcal{P})}$, $\mathcal{D}_{\Phi_1(\mathcal{P})} \subseteq \mathcal{D}_{\Phi_2(\mathcal{P})}$, $\mathcal{C}_{\Phi_1(\mathcal{P})} = \mathcal{C}_{\Phi_2(\mathcal{P})}$
(any instantiation I on $Y \subseteq X_{\Phi_2(\mathcal{P})}$ locally inconsistent in $\Phi_2(\mathcal{P})$ is locally inconsistent in $\Phi_1(\mathcal{P})$)
- Φ_1 is stricly stronger than Φ_2 if it is at least as strong as and there exists a $\mathcal{P}$: $\Phi_1(\mathcal{P}) < \Phi_2(\mathcal{P})$.
- Φ_1 and Φ_2 are incomparable if there exists a $\mathcal{P}$' and $\mathcal{P}$'' such that $\Phi_1(\mathcal{P}') < \Phi_2(\mathcal{P}')$ and $\Phi_2(\mathcal{P}'') < \Phi_1(\mathcal{P}'')$.