A Theoretical Framework for Constraint Hierarchy Solvers

Roman Barták[1]

Abstract. In the paper we propose a framework describing constraint hierarchy solvers and thus providing a theoretical foundation for construction of such solvers. The framework is based on the decomposition of the constraint hierarchy into constraint cells, i.e., chunks of constraints that are solved together. Using various decompositions we can describe a scale of constraint hierarchy solvers from the refining method to the local propagation methods. We highlight important features of the decompositions that form sufficient conditions in soundness and completeness theorems.

1 INTRODUCTION

Constraint hierarchies were introduced in [4] for description and solving of over-constrained problems. An over-constraint problem is described by a set of constraints that cannot be satisfied together unless some of the constraints are relaxed. In the constraint hierarchy each constraint *c* has assigned a label *l* describing a user preference for satisfaction of the constraint - we are speaking about a labelled constraint *c@l*. The stronger a constraint is, the more it influences the solution of the hierarchy, i.e. the more we prefer the constraint to be satisfied. Additionally, the constraint hierarchy allows "relaxing" of constraints with the same strength via weighted sum, least squares, or similar methods.

Formally, a constraint hierarchy is a set of labelled constraints. The hierarchy can be decomposed into hierarchy levels, i.e. the sets of equally preferred constraints from the hierarchy. A solution to the constraint hierarchy is a valuation of variables in the constraints such that there is no better valuation. The relation "better" is called a comparator and its main feature is respecting the hierarchy, i.e. the comparator prefers the valuations satisfying the stronger constraints. It is possible to define various comparators like locally-better or globally-better comparators, for details and precise definitions see [4,6,16].

An important aspect of constraint hierarchies is the existence of efficient satisfaction algorithms - constraint hierarchy solvers. In this paper, we are interested in two groups of these solvers: algorithms based on refining method and local propagation algorithms.

The refining algorithms satisfy first all the constraints at the strongest level and then they try to satisfy as many as possible constraints at weaker levels successively (the solution of the strongest constraints is being refined by adding weaker constraints). The refining method was first used in a simple interpreter for HCLP (Hierarchical Constraint Logic Programming) programs [6] and it is also employed in the DeltaStar algorithm [16] and in the HCLP language CHAL [14]. The refining method is general - it can be applied to any constraint hierarchy using any comparator. However, this method requires the solution to be recomputed from scratch after every change (e.g. after adding or removing a constraint).

The local propagation algorithms were designed to allow incremental changes of the constraint hierarchy and, in particular, to support fast re-computation of the solution after change of the value of a selected variable. This feature is desirable in interactive graphical applications where we need to re-compute fast the position of objects after moving the mouse. The local propagation algorithms order the constraints first (this is called planning) and then they propagate values through the constraints according to the ordering (this is called executing). Unfortunately, the local propagation algorithms lost generality so they can only be applied to special sets of constraints and comparators. The local propagation algorithms DeltaBlue [13], SkyBlue [12], QuickPlan [15], DETAIL [10], Houria [8] can solve only equality constraints, e.g., linear equations over reals. The exception is the Indigo algorithm [3] for solving inequalities that combines local propagation and refining method. We borrowed the main idea behind the Indigo algorithm, i.e., the propagation of the set of values, to our framework. The local propagation algorithms also use the locally-predicate comparator or its variant only. Only Houria III and DETAIL can use globally comparators and Indigo uses a metric comparator. Finally, local propagation is not designed to find multiple solutions easily due to the uniqueness of the propagation step (backtracking-based techniques can be used to find alternative solutions, though).

To suppress the disadvantages of both refining and local-propagation methods we generalised the local propagation algorithms or, in other words, we refined further the refining method. Instead of working with individual constraints or working with the whole hierarchy level we propose to use a decomposition of the constraint hierarchy into constraint cells and then to propagate values through the cells. In the paper, we formalise the decomposition of the constraint hierarchy into the cells and we show how to find a solution of the hierarchy using the propagation through the cells. We highlight some important features of the decompositions and we give sufficient conditions for soundness and completeness theorems. Note finally, that the proposed framework is applicable to refining algorithms (cell = a hierarchy level) as well as to local propagation algorithms (cell = a constraint). Consequently, the proposed framework provides a theoretical background for many current constraint hierarchy solvers as well as it provides a base for construction of new solvers - in [1] we described several new solvers based on this framework.

[1] Charles University, Faculty of Mathematics and Physics, Malostranské námestí 2/25, Praha, Czech Republic.
bartak@kti.mff.cuni.cz.
Supported by GACR grant 201/99/D057.

The basic structure of our framework and terminology is derived from the *generalised local propagation* by Hosobe, Matsuoka, and Yonezava [11]. Nevertheless, motivation behind our research is different from [11]. We intend to generalise local propagation in such a way that it supports arbitrary constraints (in particular, inequalities and comparisons) and comparators (including global comparators). Moreover, in addition to the soundness theorem we also prove the completeness theorem.

2 A REFORMULATION OF CONSTRAINT HIERARCHIES

To formally grasp the algorithms for solving constraint hierarchies we reformulate the definitions of constraint hierarchies while still keeping the original meaning [4]. We believe that such reformulation provides better tools for the design of effective constraint hierarchy solvers.

The basic idea behind our framework follows the concept of local propagation that is seen as the way of refining the set of valuations by satisfying the sets of constraints. In particular, rather than propagating the values through the individual constraints, we propose to propagate the valuations through the groups of constraints called constraint cells. This generalisation allows us to overcome the problems of the traditional local propagation, in particular, it is possible to solve arbitrary constraint hierarchies (including non-functional constraints like comparisons) using a wider set of comparators (including globally-better comparators). Moreover, the algorithms based on our framework can find all the solutions (for total comparators). We formulate sufficient conditions for soundness and completeness of the algorithms there.

2.1 Hierarchy comparators and hierarchy satisfiers

The notion of the comparator is crucial for the definition of the solution of the constraint hierarchy. We impose additional restrictions to the comparator so some traditional comparators cannot be used in our framework (e.g. the worst-case-better comparator). Nevertheless, all the widely used comparators are still covered by our definition. To distinguish from the original definition we are using the notion of a hierarchy comparator. The hierarchy comparator is made of level comparators that compare two valuations at a hierarchy level (or more generally, at a set of constraints without the strength annotations). We use a standard error function *e* to compare two valuations at a constraint [6,16].

Definition 1 *Given a (labelled) constraint c and a valuation* s*, the error function e indicates how nearly the constraint c is satisfied by the valuation* s *via mapping to a non-negative real number. Moreover, the error function must have the following property:*

$$e(c,\sigma)=0 \Leftrightarrow c\sigma \text{ holds.}$$

Definition 2 *Given the sets C,* C_1*, and* C_2 *of constraints, the valuations* s*,* q*, and* p*, and an error function e, we call the relation* $\overset{C}{\leq}$ *over the set of valuations a level comparator if the following conditions hold:*

a) $\sigma \overset{C}{\leq} \theta \wedge \theta \overset{C}{\leq} \pi \Rightarrow \sigma \overset{C}{\leq} \pi$

b) $(\forall c \in C \;\; e(c,\sigma) \leq e(c,\theta)) \Rightarrow \sigma \overset{C}{\leq} \theta$

c) $\theta \overset{C_1 \cup C_2}{\leq} \sigma \wedge \sigma \overset{C_1}{\leq} \theta \Rightarrow \theta \overset{C_2}{\leq} \sigma$

d) $\sigma \overset{C_1}{\leq} \theta \wedge \sigma \overset{C_2}{\leq} \theta \Rightarrow \sigma \overset{C_1 \cup C_2}{\leq} \theta$.

The conditions a) and b) of the definition describe the transitivity and "well behaviour" of the level comparator. The conditions c) and d) of the definition characterise behaviour of the level comparator when we decompose the set of constraints into two sets.

Note also, that the level comparator is defined for a particular error function *e* so we can get different level comparators for different error functions.

Now, we can define relations $\overset{C}{<}$ and $\overset{C}{\sim}$ using the level comparator $\overset{C}{\leq}$ in an obvious way:

$$\sigma \overset{C}{<} \theta \equiv_{def} \sigma \overset{C}{\leq} \theta \wedge \neg(\theta \overset{C}{\leq} \sigma) \qquad \sigma \overset{C}{\sim} \theta \equiv_{def} \sigma \overset{C}{\leq} \theta \wedge \theta \overset{C}{\leq} \sigma .$$

The hierarchy comparator, whose definition follows, compares two valuations according to the constraint hierarchy. The operational semantics of the hierarchy comparator is following. We first decompose the constraint hierarchy into the hierarchy levels and then we compare the valuations at individual hierarchy levels. Finally, we combine results of the level comparison successively (lexicographically) from the stronger to the weaker levels. Thus, the hierarchy comparator expresses explicitly the idea of respecting the hierarchy [6,16].

Definition 3 *Given a constraint hierarchy H, its hierarchy levels* H_i*, and valuations* s*,* q*, we call the relation* $\overset{H}{<}$ *over the set of valuations a hierarchy comparator if it is defined in the following way:*

$$\sigma \overset{H}{<} \theta \equiv_{def} \exists k > 0 \;\; \forall l \in \{1,\ldots,k-1\} \; \sigma \overset{H_l}{\sim} \theta \wedge \sigma \overset{H_k}{<} \theta .$$

It is also possible to define relations $\overset{H}{\sim}$ *and* $\overset{H}{\leq}$ *in an obvious way.*

When the hierarchy comparator is defined we can use it to select the valuations that satisfy best the constraint hierarchy. We call such operation a *hierarchy satisfier* - it selects the valuations from a given set of valuations to satisfy best a given constraint hierarchy.

Definition 4 *Given a set Q of valuations and a constraint hierarchy H, the function S selecting a subset from Q in the following way:*

$$S(\Theta, H) = \{\sigma \in \Theta \mid \neg\exists \theta \in \Theta \;\; \theta \overset{H}{<} \sigma\}$$

is called a hierarchy satisfier.

We identified some interesting properties of the hierarchy satisfier, the proof of them can be found in [2].

Lemma 1 *Given a hierarchy satisfier S defined using an error function e, constraint hierarchies H and H', and a set Q of valuations, the following properties hold:*

a) $\forall \sigma \in S(\Theta, H) \;\; ((\forall c \in H' \;\; e(c,\sigma) = 0) \Rightarrow \sigma \in S(\Theta, H \cup H'))$

b) $\forall \sigma, \theta \in \Theta \;\; ((\sigma \in S(\Theta, H) \wedge \sigma \overset{H}{\sim} \theta) \Rightarrow \theta \in S(\Theta, H))$.

The first feature concerns the behaviour of the hierarchy satisfier when enlarging the set of labelled constraints. It says: if the satisfier selects a valuation σ for the hierarchy H and this valuation satisfies all the constraints from the hierarchy H' then the satisfier selects this valuation for the enlarged constraint hierarchy H∪H' as well. In particular, if σ is a solution of the hierarchy H and we add a new constraint to H which is satisfied by σ then σ is still a solution of the extended constraint hierarchy.

The second feature describes the connection between the hierarchy satisfier and equivalence of valuations. It says: if we have two equally good valuations (the hierarchy comparator measures the quality) and the hierarchy satisfier selects one of these valuations then the second valuation is selected as well. In particular, if we have a solution σ of the hierarchy and any valuation θ is equally good to σ then the valuation θ is a solution too.

Now, if the set to which we apply the hierarchy satisfier contains all the valuations satisfying all the required constraints then we get exactly the solution of the constraint hierarchy. This is formally described by the following definition.

Definition 5 *Given a constraint hierarchy H and a set Q of all valuations which satisfy all the required constraints in H (i.e. all the constraints in H_0), we define the solution S(H) of the hierarchy H in the following way:*

$$S(H)=S(Q,H).$$

2.2 Decompositions of the constraint hierarchy

The original definition of the solution of the constraint hierarchy from [6] corresponds to a combination of Definition 4 and 5. We only extracted the solution step and give it a name - a hierarchy satisfier. We can apply the hierarchy satisfier to a complete constraint hierarchy to get the solution but this corresponds to solving the hierarchy as a single entity. There exist algorithms that solve the constraint hierarchy in one step, like the projection algorithm from [9], but these algorithms can hardly provide incremental behaviour similar to local propagation algorithms. Therefore it seems more desirable to decompose the constraint hierarchy into smaller sets of labelled constraints - we call them cells - and then apply the hierarchy satisfier gradually to these sets. If the constraint hierarchy is changed then we can find a new solution starting from the first changed cell instead of re-computing the solution from scratch.

Definition 6 *Given a constraint hierarchy H and a natural number $n \geq 1$, we call the sequence $B^1,\ldots,B^n$ a decomposition of H with the cells B^i if the following properties hold:*

$$\forall i \in \{1,..,n\}\ B^i \subseteq H$$
$$\forall i,j \in \{1,..,n\},\ i \neq j\ \ B^i \cap B^j = \emptyset$$
$$B^1 \cup \ldots \cup B^n = H$$

Our solution method is based on applying gradually the operator of hierarchical satisfier to this sequence. In fact, we apply this operator starting from the set of all valuations that satisfy all the required constraints in the hierarchy. First, we define what soundness of the decomposition means.

Definition 7 *Given a hierarchy satisfier S and a constraint hierarchy H, we call its decomposition $B^1,\ldots,B^n$ sound, if the following property holds for all sets Q of valuations:*

$$\boldsymbol{S(S(...S(\Theta,B^1),...),B^n) \subseteq S(\Theta,B^1 \cup ... \cup B^n) = S(\Theta,H)}$$

Typically, we will use the set *Q* of valuations satisfying the required constraints from the hierarchy. Then the above definition says that all the valuations "computed" by gradual application of the hierarchy satisfier to the sound decomposition belong to the solution set of the constraint hierarchy. It is easy to show that there exists a decomposition that is not sound.

Lemma 2 *There exists a constraint hierarchy H that has a non-sound decomposition.*

Proof Let A={x=1@weak} and B={x=2@strong} and A,B is a decomposition of H. Then visibly:

$$\boldsymbol{S(A \cup B) = \{\{x/2\}\}}$$
$$\boldsymbol{S(S(A),B) = S(\{\{x/1\}\},B) = \{\{x/1\}\}}$$ ❑

The problem highlighted in the proof of Lemma 2 is that there is a constraint in the first cell (x=1@weak) whose satisfaction causes relaxation of a stronger constraint (x=2@strong) in the subsequent cell. We should ensure that this situation never occurs in the sequence of the cells, otherwise we cannot guarantee soundness of the method. Using the above observation, we formulate a sufficient condition - a gradual weakening property - for the sequence of cells that guarantees soundness of the decomposition. It says the following: if some valuation is selected by the hierarchy satisfier and any constraint is violated by this valuation then all the constraints in the previous cells must be stronger than the strongest constraint in the cell where the violated constraint is located.

Definition 8 *We say that the sequence of cells $B^1,\ldots,B^n$ satisfies the gradual weakening property if the following implication holds for every $i \in \{1,\ldots,n-1\}$ and for every set Q of valuations:*

$$\boldsymbol{(\exists \sigma \in S(S(...S(\Theta,B^1),...),B^{i+1}) \cup S(\Theta,B^1 \cup ... \cup B^{i+1})}$$
$$\boldsymbol{\exists c@l \in B^{i+1}\ \ e(c,\sigma) > 0)}$$
$$\boldsymbol{\Rightarrow (\forall j \leq i\ \ \forall c'@k' \in B^j\ \ \forall c''@k'' \in B^{i+1}\ \ k' < k'')}$$

The above definition of the gradual weakening property describes formally the idea behind the constraint hierarchies. We prefer satisfaction of a stronger constraint to satisfaction of an arbitrary number of weaker constraints, or, in other words, dissatisfaction of the constraint is not caused by satisfaction of a weaker constraint.

There exists a trivial sound decomposition of the constraint hierarchy into hierarchy levels, i.e., $B^i = H_i$. Visibly, this decomposition satisfies the gradual weakening property because if $i<j$ then all the constraints in B^i are stronger than the constraints in B^j. The refining algorithms use such decomposition, so our approach covers the refining method for solving constraint hierarchies (and the theory proves that the refining method is sound). Basically, the solving algorithm is more efficient and incremental if it uses a finer decomposition, i.e., the decomposition into smaller cells. In [1] we showed some algorithms for construction of

decompositions finer than the decomposition into hierarchy levels.

The following auxiliary lemma shows some features of the gradual weakening property. This lemma can be seen as an extension of Lemma 1 to a sequence of cells. In particular, if the method of solving constraint hierarchy (by decomposition into cells and application of the hierarchy satisfier) is sound and it provides some valuation σ which is equally good to another valuation θ then this second valuation θ is computed by the method as well.

Lemma 3 *Let q be a valuation in Q and $B^1,\ldots,B^n$ be a sequence of cells that satisfies the gradual weakening property.*

If $\sigma \in S(S(\ldots S(\Theta, B^1),\ldots), B^n)$ such that $\sigma \overset{B^1\cup\ldots\cup B^n}{\sim} \theta$ *and* $\forall i \in \{1,\ldots,n\}\ S(S(\ldots S(\Theta, B^1),\ldots), B^i) \subseteq S(\Theta, B^1 \cup \ldots \cup B^i)$ *then the following formulas hold:*

a) $\forall i \in \{1,\ldots,n\}\ \sigma \overset{B^1\cup\ldots\cup B^i}{\sim} \theta$

b) $\forall i \in \{1,\ldots,n\}\ \theta \in S(S(\ldots S(\Theta, B^1),\ldots), B^i)$.

The gradual weakening property is a sufficient condition that guarantees the soundness of algorithms for solving constraint hierarchies based on the decomposition of the hierarchy into cells and gradual application of the hierarchy satisfier.

Theorem 4 *Let $B^1,\ldots,B^n$ be a sequence of cells satisfying the gradual weakening property. Then the following formula holds:*

$$S(S(\ldots S(\Theta, B^1),\ldots), B^n) \subseteq S(\Theta, B^1 \cup \ldots \cup B^n).$$

Proof: By induction of the length of the sequence of cells.

Base: $S(\Theta,B^1) \subseteq S(\Theta,B^1)$

Induction step: assume that the following formula holds

$$\forall i \in \{1,\ldots n\}\ S(\ldots S(\Theta, B^1),\ldots, B^i) \subseteq S(\Theta, B^1 \cup \ldots \cup B^i)$$

Assume for contradiction

$$\exists s \in S(\ldots S(\Theta, B^1),\ldots, B^{n+1}) \text{ s.t. } s \notin S(\Theta, B^1 \cup \ldots \cup B^{n+1}) \quad \textbf{(1)}$$

From the induction assumption and (1):

$$s \in S(\Theta, B^1 \cup \ldots \cup B^n) \quad \textbf{(2)}$$

Case "c@l Î B^{n+1} e(c,s)=0

From (2) and Lemma 1 a): $s \in S(\Theta, B^1 \cup \ldots \cup B^{n+1})$, which is a contradiction with (1)

Case $c@l Î B^{n+1} e(c,s)>0

From the gradual weakening property: B^{n+1} contains constraints weaker than the constraints in the set $B^1 \cup \ldots \cup B^n$; let $m = \min\{l \mid c@l \in B^{n+1}\}$, i.e., *m* is the strongest label among the constraints in B^{n+1}

from (1), Definition 2, and Definition 3:

$$\exists p \in \Theta\ p \overset{B^1\cup\ldots\cup B^{n+1}}{<} s \text{ i.e.}$$

$$\exists k > 0\ \forall l \in \{1,\ldots k-1\}\ p \overset{(B^1\cup\ldots\cup B^{n+1})_l}{\sim} s \wedge p \overset{(B^1\cup\ldots\cup B^{n+1})_k}{<} s \quad \textbf{(3)}$$

Case k<m, i.e., the valuations σ and π are distinguished at the level stronger than the strongest constraint in B^{n+1}

thus $\forall j \le k\ (B^1\cup\ldots\cup B^{n+1})_j = (B^1\cup\ldots\cup B^n)_j$ because $B^{n+1}_j = \varnothing$

together with (3): $p \overset{B^1\cup\ldots\cup B^n}{<} s$ i.e.,

$s \notin S(\Theta, B^1 \cup \ldots \cup B^n)$, which is a contradiction with (2)

Case k³m

From the gradual weakening property:

$\forall j < m\ (B^1 \cup \ldots \cup B^{n+1})_j = (B^1 \cup \ldots \cup B^n)_j$ and

$\forall j \ge m\ (B^1 \cup \ldots \cup B^n)_j = \varnothing$

together with (3): $p \overset{B^1\cup\ldots\cup B^n}{\sim} s$

from Lemma 3 b): $p \in S(\ldots S(\Theta, B^1), \ldots, B^n)$

from the gradual weakening property and (3): $p \overset{B^{n+1}}{<} s$

thus $s \notin S(\ldots S(\Theta, B^1), \ldots, B^{n+1})$ which is a contradiction with (1)

We showed that the assumption (1) leads to a contradiction so the following formula holds:

$$\forall s\ (s \in S(\ldots S(\Theta, B^1), \ldots, B^{n+1}) \Rightarrow s \in S(\Theta, B^1 \cup \ldots \cup B^{n+1})), \text{ i.e.,}$$

$$S(\ldots S(\Theta, B^1), \ldots, B^{n+1}) \subseteq S(\Theta, B^1 \cup \ldots \cup B^{n+1}).$$

❑

Theorem 4 guarantees soundness of the proposed method for solving constraint hierarchies, i.e., all the valuations found by applying gradually the hierarchy satisfier into the sequence of cells satisfying the gradual weakening property (GWP) belong to the solution set of the constraint hierarchy. Note also, that GWP is a feature of the decomposition into cells so it is independent of the problem. It means that any constraint hierarchy (any problem) can be solved using the above described method.

Sometimes, one needs to find all the valuations from the solution set. Hence, we looked for an additional condition(s) that guarantees the completeness of the algorithm. We found that such a condition is linearity of the hierarchy comparator, i.e., the hierarchy comparator totally orders all the valuations according to a given constraint hierarchy. We call such a comparator a total comparator.

Definition 9 *We say that the level comparator* $\overset{C}{\le}$ *is a total level comparator if the following condition holds:*

$$\forall \sigma, \theta\ \ \sigma \overset{C}{\le} \theta \vee \theta \overset{C}{\le} \sigma.$$

Definition 10 We say that the hierarchy comparator $\overset{H}{\le}$ is a *total hierarchy comparator* if it is defined using a total level comparator.

Notice that any globally-better comparator, that is a hierarchy comparator, is a total hierarchy comparator. However, the locally-better comparators are not total hierarchy comparators in general because two valuations might be incomparable (e.g. H = {x=1@weak, x=2@weak}, σ = {x/1}, θ = {x/2}).

We prove below that our solving method may provide all the solutions if the comparator is total. Opposite to GWP (the soundness condition) which impose no restriction to the problem, using the total comparator is a stronger condition that is not satisfied by some problems/hierarchies. Nevertheless, when the user requires all the solutions then he or she may use global comparators and in such a case the method guarantees completeness.

The following lemma shows the completeness of a single propagation step. It is a simplified version of the full

completeness theorem restricted to the decomposition consisting of two cells.

Lemma 5 *Let B^1, B^2 be two cells satisfying the gradual weakening property, S be a hierarchy satisfier that is defined using a total hierarchy comparator and* $S(S(\Theta, B^1), B^2) \neq \emptyset$. *Then the following formula holds:*

$$S(S(\Theta, B^1), B^2) = S(\Theta, B^1 \cup B^2).$$

Theorem 6 *Let $B^1,\ldots,B^n$ be a sequence of cells satisfying the gradual weakening property, S be a hierarchy satisfier that is defined using a total hierarchy comparator and* $S(S(...S(\Theta, B^1),...), B^n) \neq \emptyset$. *Then the following formula holds:*

$$S(S(...S(\Theta, B^1),...), B^n) = S(\Theta, B^1 \cup ... \cup B^n).$$

Proof: By induction of the length of the sequence of cells.
Base: $S(\Theta, B^1) = S(\Theta, B^1)$
Induction step: Assume that the following equality holds:

$$S(S(...S(\Theta, B^1),...), B^n) = S(\Theta, B^1 \cup ... \cup B^n)$$

Thus

$$S(S(...S(\Theta, B^1),..., B^n), B^{n+1}) = S(S(\Theta, B^1 \cup ... \cup B^n), B^{n+1}) \neq \emptyset$$

By Lemma 5

$$S(S(\Theta, B^1 \cup ... \cup B^n), B^{n+1}) = S(\Theta, B^1 \cup ... \cup B^n \cup B^{n+1})$$

Together:

$$S(S(...S(\Theta, B^1),..., B^n), B^{n+1}) = S(\Theta, B^1 \cup ... \cup B^n \cup B^{n+1})$$

❑

Note that the completeness theorem requires one more condition to be satisfied, in particular $S(S(...S(\Theta, B^1),...), B^n) \neq \emptyset$, i.e., the proposed solution method has to find at least one valuation.

3 CONCLUSIONS

In the paper we propose a theoretical framework for solving constraint hierarchies by decomposition into constraint cells. We also suggest sufficient conditions for soundness and completeness of the solving methods based on this framework (see [2] for proofs of all the lemmas).

The solvers based on our framework first decompose the hierarchy into cells in such a way that this decomposition satisfies the gradual weakening property. Then they propagate valuations through these cells. In the worst case, the decomposition into hierarchy levels might be used. However, better behaviour of the solver can be achieved by using finer decomposition. Current local propagation algorithms can decompose some hierarchies into single constraints but these algorithms have many limitations (see Introduction). In [1], we propose a general decomposition algorithm that can be applied to any constraint hierarchy.

The major contribution of this paper is providing a theoretical framework for description of existing solvers as well as for design of new solvers based on decomposition into cells. We expect that these upcoming constraint hierarchy solvers will handle non-functional constraints as well as global comparators while still preserving advantages of local propagation like incremental updates of the constraint hierarchy or fast re-computing of the solution after change of the value of some variable.

REFERENCES

[1] Barták R.: Constraint Hierarchy Networks, in Proceedings of 3rd ERCIM/CompulogNet Workshop on Constraints, Amsterdam, 1998.

[2] Barták, R.: Expert Systems Based on Constraints, PhD. Thesis, Charles University, 1997 (http://kti.mff.cuni.cz/~bartak).

[3] Borning, A., Anderson, R., Freeman-Benson, B., Indigo: A Local Propagation Algorithm for Inequality Constraints, in: *Proceedings of the 1996 ACM Symposium on User Interface Software and Technology*, pp. 129-136, 1996.

[4] Borning, A., Duisberg, R., Freeman-Benson, B., Kramer, A., Woolf, M., Constraint Hierarchies, in: *Proceedings of the 1987 ACM Conference on Object Oriented Programming Systems, Languages, and Applications*, pp.48-60, 1987.

[5] Borning, A., Freeman-Benson, B., The OTI Constraint Solver: A Constraint Library for Constructing Interactive Graphical User Interfaces, in: *Principles and Practice of Constraint Programming - CP95 (U.Montanari, F.Rossi eds.)*, pp. 624-628, Cassis, France, 1995.

[6] Borning, A., Maher, M., Martindale, A., Wilson, M., Constraint Hierarchies and Logic Programming, in: *Proceedings of the Sixth International Conference on Logic Programming*, pp. 149-164, Lisbon, 1989.

[7] Borning A., Marriott K., Stuckey P., and Xiao Y., Solving Linear Arithmetic Constraints for User Interface Applications, in *Proceedings of the 1997 ACM Symposium on User Interface Software and Technology*, pp. 87-96, 1997.

[8] Bouzoubaa, M., Neveu, B., Hasle, G., Houria III: Solver for Hierarchical System, Planning of Lexicographic Weight Sum Better Graph For Functional Constraints, in: *the Fifth INFORMS Computer Science Technical Section Conference on Computer Science and Operations Research*, Dallas, Texas, 1996.

[9] Harvey, W., Stuckey, P.J., Borning, A., Compiling Constraint Solving Using Projection, in: *Principles and Practice of Constraint Programming - CP'97 (G. Smolka ed.)*, pp. 491-505, Springer-Verlag, 1997.

[10] Hosobe, H., Miyashita, K., Takahashi, S., Matsuoka, S., Yonezawa, A., Locally Simultaneous Constraint Satisfaction, in: *Principles and Practice of Constraint Programming - PPCP'94 (A. Borning ed.)*, pp. 51-62, Springer-Verlag, 1994.

[11] Hosobe, H., Matsuoka, S., Yonezawa, A., Generalized Local Propagation: A Framework for Solving Constraint Hierarchies, in: *Principles and Practice of Constraint Programming - CP'96 (E. Freuder ed.)*, pp. 237-251, Springer-Verlag, 1996.

[12] Sannella, M., The SkyBlue Constraint Solver, Tech. Report 92-07-02, Department of Computer Science and Engineering, University of Washington, 1993.

[13] Sannella, M., Freeman-Benson, B., Maloney, J., Borning,A., Multi-way versus One-way Constraints in User Interfaces: Experience with the DeltaBlue Algorithm, in Software--Practice and Experience, Vol. 23 No. 5, pp. 529-566, 1993.

[14] Satoh K., Aiba A., Computing Soft Constraints by Hierarchical Constraint Logic Programming, in: Journal of Information Processing, 7, 1993.

[15] Vander Zanden, B., An Incremental Algorithm for Satisfying Hierarchies of Multi-way Dataflow Constraints, Tech. Report, Department of Computer Science, University of Tennessee, 1995.

[16] Wilson, M., Borning, A., Hierarchical Constraint Logic Programming, in: The Journal of Logic Programming, special issue on Constraint Logic Programming, Vol. 16 No. 3 & 4, pp. 227-318, 1993.