ICOT Technical Report: TR-610

TR-610

Computing Soft Constraints by Hierarchical Constraint Logic Programming

by
K. Satoh & A. Aiba

January, 1991

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

Computing Soft Constraints by Hierarchical Constraint Logic Programming

Ken Satoh and Akira Aiba

Institute for New Generation Computer Technology (ICOT)

4-28 Mita 1-Chome Minato-ku, Tokyo 108 Japan

phone: +81-3-3456-2514, email: ksatoh@icot.or.jp

Abstract

We have formalized soft constraints in [11] based on interpretation ordering which is a generalization of circumscription. However, this formalization is written in a second-order formula and therefore is not computable in general. To make it computable, we have to introduce some restriction.

In this paper, we propose a *semantic restriction*. By semantic restriction, we mean that we fix the considered domain so that interpretations of domain-dependent relations are fixed, and soft and hard constraints contain only domain-dependent relations. If we accept this restriction, the soft constraints can be expressed in a first-order formula. Moreover, there is already a proposed mechanism suitable for computing such restricted soft constraints in the literature, that is, hierarchical constraint logic programming languages (HCLP languages) [2, 9].

Firstly, we identify a solution to constraint hierarchy defined by HCLP languages with the most preferable solution for semantically-restricted soft constraints. Then, we provide an algorithm for calculating all the most preferable solutions for soft constraints without any redundant calls of constraint solver. Then, we show examples of computing soft constraints by using our HCLP language named CHAL(Contraintes Hierarchiques avec Logique) [9].

Keywords: constraint logic programming, soft constraint, preference, constraint hierarchy, interpretation ordering, circumscription

1 Introduction

In the area of *synthesis* problems such as job shop scheduling, circuit design and planning, there are two kinds of constraints. One kind is *hard constraints* which every solution is required to satisfy and the other is *soft constraints* which provide preferences over solutions [4, 12]. Most systems manipulating soft constraints use evaluation functions to represent these soft constraints. However, it is hard to debug evaluation functions if obtained solutions are unsatisfactory. One possible solution to this problem is to manipulate soft constraints in a logical manner.

We have proposed a logical foundation of soft constraints in [11] by using a meta-language [10] which expresses an interpretation ordering. The idea of formalizing soft constraints is as follows. Let hard constraints be represented in first-order formulas. Then an interpretation which satisfies all of these first-order formulas can be regarded as a possible solution and soft constraints can be regarded as an order over those interpretations because soft constraints represent criteria over possible solutions to choose the most preferable ones. We use a meta-language which represents a preference order directly. This meta-language can be translated into a second-order formula to provide a syntactical definition of the most preferable solutions.

Although this framework is rigorous and declarative, it is not computable in general because it is defined by a second-order formula. Therefore, we have to restrict a class of constraints so that these constraints are computable.

Since the above interpretation ordering is a generalization of circumscription, one might think that restrictions for computing circumscription can be helpful. However, previous proposals of restricting circumscription are domain-independent, that is, syntactical such as separable axioms [5] for ordinary circumscription or stratified axioms [6] for prioritized circumscription. Although this technique is applicable in any domain, this restriction is not so clear at identifying useful applications.

In this paper, we propose another restriction based on semantics. By *semantic restriction*, we mean that we fix the considered domain so that interpretations of domain-dependent relations are fixed, and soft and hard constraints consist of only

domain-dependent relations. This restriction is fairly reasonable because when we solve any actual problem with soft constraints such as design and planning, we usually know the domain of the problem. If we accept this restriction, the soft constraints can be expressed in a first-order formula. Moreover, there is already a proposed mechanism suitable for computing such restricted soft constraints in the literature, that is, hierarchical constraint logic programming languages (HCLP languages) [2, 9].

In the following section, we show a relationship between semantics of soft constraints in [11] and constraint hierarchy defined by HCLP language. By this relationship, we use HCLP to compute the most preferable solutions specified by semantically-restricted soft constraints.

Then, we show an algorithm to compute all simplified constraint sets of the most preferable solutions from constraint hierarchy without any redundant calls of constraint solver and show examples of computing soft constraints by our HCLP language CHAL.

2 Constraint Hierarchy and Soft Constraints

In this section, we define HCLP and then, show a relationship between constraint hierarchy defined by HCLP and the soft constraints proposed in [11].

We follow the definition of HCLP in [2] but extend it by introducing a complex form of soft constraints. HCLP language is a language augmenting CLP language with labeled constraints. An HCLP program consists of rules of the form:

$$h :- b_1, ..., b_n$$

where $h, b_1, ..., b_n$ are predicates or constraints or labeled constraints. Labeled constraints is of the form:

label C

where C is a complex constraint in which only domain-dependent functional symbols can be allowed as functional symbols and **label** is a label which expresses strength of the complex constraint C.

A complex constraint is a disjunction of conjunctions of domain-dependent con-

straints of the form:

$$((c_{11}, c_{12}, ..., c_{1n_1}); (c_{21}, c_{22}, ..., c_{2n_2}); ...; (c_{m1}, c_{m2}, ..., c_{mn_m}))$$

where $1 \leq m$ and $0 \leq n_i$ and ';' expresses a disjunction and ',' expresses a conjunction and c_{ij} is an atomic constraint whose constraint symbol is domain-dependent.

The operational semantics for HCLP is similar to CLP except manipulating constraint hierarchy. In HCLP, we accumulate labeled constraints to form constraint hierarchy by each label while executing CLP until CLP solves all goals and gives a reduced required constraints. Then, we solve constraint hierarchy with required constraints.

An assignment of variables is a mapping from each variables in constraints to an element of the considered domain. We say an assignment θ satisfies a constraint if the substitution of the free variables in the constraint makes the constraint true. An assignment of variables are partially ordered by the locally-predicate-better comparator as follows.

Let θ and σ be assignments and C_θ^1(and C_σ^1) be a set of constraints in the strongest level of the hierarchy satisfied by θ(and σ), and C_θ^2(and C_σ^2) be a set of constraints in the second strongest level of the hierarchy satisfied by θ(and σ),...., and C_θ^k(and C_σ^k) be a set of constraints in the k-th strongest level of the hierarchy satisfied by θ(and σ). θ is *locally-predicate-better* than σ w.r.t. the constraint hierarchy if

there exists $i(1 \leq i \leq k)$ such that $C_\sigma^i \subset C_\theta^i$ if for every $j(1 \leq j \leq i-1), C_\sigma^j = C_\theta^j$ where $\subset$ expresses a strict subset relation.

Then, a *solution w.r.t. the required constraints and the constraint hierarchy* is defined as an assignment θ which satisfies the required constraints and has no assignment σ such that σ satisfies the required constraints and σ is locally-predicate-better than θ.

Now, we relate solutions w.r.t. required constraints and a constraint hierarchy with the most preferable solutions for soft constraints with priority defined in [11]. We regard an assignment of variables as an interpretation and the order defined by the locally-predicate-better comparator as an order over those interpretations.

Let A be the axioms of the considered domain D, and M' and M be interpretations

with the domain D each of which satisfies A and differs from the other interpretation in at most the assignments of free variables in constraints. Since M' and M differs in at most the assignments of free variables, there is one-to-one correspondence between an interpretation and the above assignment of free variables. We say a complex constraint C *is satisfied by* an interpretation M (written as $M \models C$) if the following condition is satisfied.

1. If C is of the form $c(t_1, ..., t_l)$ where c is a constraint symbol and $t_i (1 \leq i \leq l)$ is a term then $\langle t_1^M, ..., t_l^M \rangle \in c^M$ where $t_i^M (1 \leq i \leq l)$ is an element of D to which is mapped from t_i by M and c^M is a subset of D^l to which is mapped from a constraint symbol c by M.

2. If C is of the form $(c_1, ..., c_n)$ where $c_i (1 \leq i \leq n)$ is an atomic constraint, then for every $i (1 \leq i \leq n)$ $M \models c_i$.

3. If C is of the form $(D_1; ...; D_m)$ where $D_i (1 \leq i \leq m)$ is a conjunction of atomic constraints then there exists $i (1 \leq i \leq m)$ such that $M \models D_i$.

Let $C_1^1, ..., C_{m_1}^1$ be constraints in the strongest level of the hierarchy and $C_1^2, ..., C_{m_2}^2$ be constraints in the second strongest level of the hierarchy,..., and $C_1^k, ..., C_{m_k}^k$ be constraints in the k-th strongest level of the hierarchy. Now, we define an order over interpretations by the above constraint hierarchy. We say M' *is preferred to* M (written as $M' < M$) if

$M' \leq M$ is true and $M \leq M'$ is not true

where $M' \leq M$ if for every $\forall i (1 \leq i \leq k)(M' \leq^i M)$ is true and $M' \leq^i M$ is defined as follows:

If $\forall j (1 \leq j \leq i-1) \forall l (1 \leq l \leq m_j)(M \models C_l^j$ iff $M' \models C_l^j)$ then
$\forall l (1 \leq l \leq m_i)$ if $M \models C_l^i$ then $M' \models C_l^i$.

We can easily see that this order is equivalent to an order defined by locally-predicate-better comparator from the correspondence between an interpretation and an assignment of variables.

Let A be the axioms of the considered domain. Then, the *most preferable solutions w.r.t required constraints and the order* $<$ is a model M which satisfies A and the required constraints and there is no model M' such that M' satisfies A and the required constraints and $M' < M$. From the equivalence between an order defined by locally-predicate-better comparator and the order $<$, the most preferable solutions w.r.t. required constraints and the order $<$ are equivalent to the solutions w.r.t. required constraints and constraint hierarchy.

We can give a syntactic definition of the most preferable solutions from the result in [11]. Let A be the axioms of the considered domain and $\mathbf{x}$ be a tuple of all free variables contained in required constraints and soft constraints and $RC(\mathbf{x})$ be a conjunction of required constraints and $C_1^i(\mathbf{x}), ..., C_{m_i}^i(\mathbf{x})$ $(i = 1, ..., k)$ be soft constraints. Then, the following is a syntactic definition of the most preferable solutions.

$$A \wedge RC(\mathbf{x}) \wedge \neg \exists \mathbf{y}(RC(\mathbf{y}) \wedge (\mathbf{y} \leq \mathbf{x}) \wedge \neg(\mathbf{x} \leq \mathbf{y})) \qquad \text{(P)}$$

where $\mathbf{y} \leq \mathbf{x}$ is an abbreviation of $(\mathbf{y} \leq^1 \mathbf{x}) \wedge ... \wedge (\mathbf{y} <^k \mathbf{x})$ and $\mathbf{y} \leq^i \mathbf{x}$ is an abbreviation of the following formula:

$$(\bigwedge_{j=1}^{i-1} \bigwedge_{l=1}^{m_j} (C_l^j(\mathbf{x}) \equiv C_l^j(\mathbf{y}))) \supset (\bigwedge_{l=1}^{m_i} (C_l^i(\mathbf{x}) \supset C_l^i(\mathbf{y}))).$$

Adapted from the result in [11], we can show the following theorem.

Theorem 1 *M is a most preferable solution w.r.t $RC(\mathbf{x})$ and the order $<$ if and only if M is a model of the formula (P).*

Although the general definition of the most preferable solution in [11] is written in a second-order formula, the above formula is a first order formula, that is, computable. It is because each of the above constraints is a logical combination of domain-dependent constraints with a fixed interpretation and therefore, only parameters in the above formula are free variables in constraints.

From the point of view of soft constraints in [11], we can regard this restriction as a *semantic restriction* because we fix the considered domain and use only domain-dependent constraints. This restriction is fairly reasonable because when we solve

actual problems with soft constraints such as design and planning, we usually know the domain of the problem.

Now, we show a relationship between the most preferable solutions and maximal consistent sets w.r.t. required constraints and constraint hierarchy. This is the key relation to calculate the most preferable solutions in HCLP languages.

Firstly, we define *maximal consistent set* of constraints.

Definition 1 *Let A be tha axioms of the considered domain and $RC(\mathbf{x})$ be a conjunction of required constraints and $CH(\mathbf{x})$ be a constraint hierarchy with k levels. A set of constraints $MC(\mathbf{x})$ is* maximal consistent *w.r.t A and $RC(\mathbf{x})$ and $CH(\mathbf{x})$ if $MC(\mathbf{x})$ satisfies the following conditions.*

1. *$MC(\mathbf{x})$ is consistent with A.*
2. *$MC(\mathbf{x})$ is logically equivalent to $RC(\mathbf{x}) \wedge CH^1(\mathbf{x}) \wedge ... \wedge CH^k(\mathbf{x})$ where $CH^i(\mathbf{x})$ is a conjunction of some constraints in the i-th level of $CH(\mathbf{x})$.*
3. *There is no consistent set of constraints $MC'(\mathbf{x})$ with A which is logically equivalent to $RC(\mathbf{x}) \wedge CH'^1(\mathbf{x}) \wedge ... \wedge CH'^k(\mathbf{x})$ where $CH'^i(\mathbf{x})$ is a conjunction of some constraints in the i-th level of $CH(\mathbf{x})$ and satisfies the following condition:*

   There exists $i (1 \leq i \leq k)$ such that $CH^i(\mathbf{x}) \subset CH'^i(\mathbf{x})$
   if for every $j (1 \leq j \leq i-1), CH^j(\mathbf{x}) = CH'^j(\mathbf{x})$.

Then, a relation between maximal consistent sets and the most preferable solutions is as follows.

Theorem 2 *Let A be tha axioms of the considered domain and $RC(\mathbf{x})$ be a conjunction of required constraints and $CH(\mathbf{x})$ be a constraint hierarchy. Let $MC_1(\mathbf{x}), MC_2(\mathbf{x}), ..., MC_n(\mathbf{x})$ be all maximal consistent sets w.r.t. A and $RC(\mathbf{x})$ and $CH(\mathbf{x})$. Then, $A \wedge (MC_1(\mathbf{x}) \vee MC_2(\mathbf{x}) \vee ... \vee MC_n(\mathbf{x}))$ is logically equivalent to the formula (P). In other words, models of $A \wedge (MC_1(\mathbf{x}) \vee MC_2(\mathbf{x}) \vee ... \vee MC_n(\mathbf{x}))$ are exactly all the most preferable solutions.*

If we have a *satisfaction-complete* constraint solver which can determine whether a set of constraints is consistent or not, then we neither need to write the first-order axioms of the domain nor need to use first-order inference rules to infer the most preferable solutions. All we have to do is to write constraints directly in HCLP and use the satisfaction-complete solver to compute all maximal consistent sets by generating every consistent subset of the constraint hierarchy and checking if it is maximal.

3 Algorithm to Solve Constraint Hierarchy

We show an algorithm for solving constraint hierarchy in Appendix A. Inputs of the algorithm are a constraint hierarchy and a set of reduced required constraints and its output is all maximal consistent sets of constraints simplified by the constraint solver. Features of this algorithm are as follows.

1. There is no redundant calls of constraint solver for the same combination of constraints since it calculates reduced constraints in bottom-up manner.

2. If an inconsistent combination of constraints is found by calling constraint solver, it is registered as a nogood and used for further contradiction detecting and any extension of the combination will not be processed to avoid vain combinations.

3. Inconsistency is detected without a call of constraint solver if a processed combination subsumes a registered nogood.

In [2], Borning et al. give an algorithm of solving constraint hierarchy. However, it uses backtrack to get an alternative solution and so, it may call the constraint solver for the same combination of constraints redundantly.

Although our algorithm has no redundant calls of constraint solver, it will call the constraint solver $2^n - 1$ times in the worst case where n is a number of soft constraints. So, we must ensure that inconsistency occurs at a small combination of constraints or we must prioritize constraints almost linearly. If we linearize constraints completely, then our algorithm will call constraint solver only n times.

Our algorithm has been implemented already on PSI (Personal Sequential Inference) Machine developed in ICOT. By using the algorithm, we have implemented an HCLP language called CHAL (Contraintes Hierarchiques avec Logique) [9], an extension of CAL (Contraintes avec Logique) [7] which is a CLP language developed in ICOT. In CHAL, we can use the following constraint solvers in CAL. One is an algebraic constraint solver which manipulates multi-variate polynomial equations based on Buchberger algorithm [3] to calculate Gröbner bases and the other is a Boolean constraint solver which extends Buchberger algorithm to handle propositional Boolean equations [8].

4 Examples

Now, we show two CHAL examples of calculating the most preferable solutions. One is a meeting scheduling problem solved by Boolean CHAL and the other is a multi-axis gearbox design problem solved by algebraic CHAL.

4.1 Meeting Scheduling Problem

In this subsection, we show an example of solving meeting scheduling problem in [11] by using Boolean CHAL.

In a Boolean CHAL program, we express constraints as Boolean equations and in Boolean equations, we can use the only constraint symbol, = and the function symbols such as /\ (conjunction), \/ (disjunction), -> (implication), <-> (equivalence), ~ (negation) and the constants such as 1 as truth and 0 as falsity and propositional variables.

Firstly, we regard the following terms as propositional variables. $c(x)$ represents that the meeting will be held on day x and $p(x), v(x), m(x)$ represent that the president, the vice president and the manager attend the meeting on day x.

Suppose that we consider a meeting schedule for day 1, 2 and 3 and the following hard constraints represented in Boolean equations exist.

1. The meeting must be held:

$(c(1) \backslash / c(2) \backslash / c(3)) = 1$.

2. The president must attend the meeting:

   $(c(x)\text{->}p(x)) = 1$ for all $x = 1, 2, 3$.

3. Since $p(x)$ expresses that the president attends the meeting on day x, if it is true, $c(x)$ (the meeting is held on day x) is also true:

   $(p(x)\text{->}c(x)) = 1$ for all $x = 1, 2, 3$.

4. The same thing holds if the vice president or the manager attends the meeting.

We can expand these constraints as follows:

$(v(x)\text{->}c(x)) = 1$ for all $x = 1, 2, 3$ and $(m(x)\text{->}c(x)) = 1$ for all $x = 1, 2, 3$.

5. The president cannot attend the meeting on day 1 and the manager cannot attend the meeting on day 2:

$p(1) = 0$ and $m(2) = 0$

And we consider the following soft constraints.

1. The vice president should *preferably* attend the meeting. This soft constraint means that $(c(x)\text{->}v(x)) = 1$ should be satisfied as much as possible for all $x = 1, 2, 3$.

2. The manager also should *preferably* attend the meeting. This soft constraint means that $(c(x)\text{->}m(x)) = 1$ should be satisfied as much as possible for all $x = 1, 2, 3$

3. The schedule of the vice president is *prioritized* to the schedule of the manager. This priority means that $(c(x)\text{->}v(x)) = 1$ is stronger than $(c(y)\text{->}m(y)) = 1$ for every $x = 1, 2, 3$ and $y = 1, 2, 3$. To do so, we attach the stronger label to $(c(x)\text{->}v(x)) = 1$ than to $(c(y)\text{->}m(y)) = 1$ for every $x = 1, 2, 3$ and $y = 1, 2, 3$.

Then, a Boolean CHAL program which builds constraint hierarchy of the above example is shown as follows.

```
meeting1 :-
    bool:(c(1)\/c(2)\/c(3))=1,hard([1,2,3]),soft([1,2,3]),
    bool:p(1)=0,bool:m(2)=0.
```

```
meeting2 :- meeting1,bool:v(3)=0.
hard([]).
hard([X|Y]):-
    bool:(c(X)<->p(X))=1,bool:(v(X)->c(X))=1,bool:(m(X)->c(X))=1,hard(Y).
soft([]).
soft([X|Y]):-
    chal:soft(bool:(c(X)->v(X))=1,0),chal:soft(bool:(c(X)->m(X))=1,1),soft(Y).
```

In the above program,

`chal:soft(bool:(c(X)->v(X))=1,0)` and `chal:soft(bool:(c(X)->m(X))=1,1)` express soft constraints. The first argument is a constraint and the second argument expresses the strength of the constraint. In CHAL, we use a natural number to express the strength. A soft constraint with the number 0 is the strongest and a constraint becomes weaker as the associated number becomes bigger.

If we ask `?-meeting1`, then Boolean CHAL firstly calculates a set of reduced constraints from required constraints and then computes all simplified maximal consistent sets of constraints by solving constraint hierarchy. In this case, Boolean CHAL returns only one maximal consistent set which includes $c(1) = 0, c(2) = 0, c(3) = 1$. Since on day 3, all of three can attend the meeting, day 3 is selected for the most preferable date for the meeting.

The conclusion may be withdrawn by adding another constraint. For example, suppose a new constraint that the vice president cannot attend the meeting on day 3 is added. That is, the following constraint is added:

$v(3) = 0$.

This is done by asking `?-meeting2`, and Boolean CHAL returns only one maximal consistent set which includes $c(1) = 0, c(2) = 1, c(3) = 0$. This means that day 2 is the most preferable meeting date in this new situation because the schedule of the vice president has the priority to the schedule of the manager. This expresses nonmonotonic character of soft constraints.

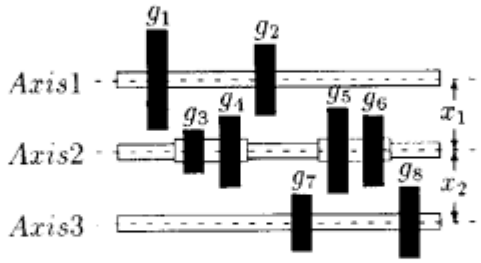

Figure 1: Mutli-Axis Gearbox

4.2 Gear Design

In this subsection, we show an example in algebraic CHAL adapted from a design problem for a multi-axis gearbox. A gearbox is used to produce various speeds from the main spin. Figure 1 shows an example of a three-axis gearbox.

Gears on $Axis1$ (g_1, g_2 in Figure 1) and $Axis3$ (g_7, g_8 in Figure 1) are fixed and gears on $Axis2$ (g_3, g_4, g_5, g_6 in Figure 1) are slidable. Slidable gears slide along the axis and mesh with fixed gears. In Figure 1, each pair of $\langle g_1, g_3 \rangle$, $\langle g_2, g_4 \rangle$, $\langle g_5, g_7 \rangle$, $\langle g_6, g_8 \rangle$, can mesh. Since there are two gear changes between $Axis1$ and $Axis2$, and two gear changes between $Axis2$ and $Axis3$, there are totally four output speeds ($= 2 * 2$) by combinations of those gear changes.

Here, we consider the following design problem for a three-axis gearbox. We specify the sum of the two intervals between axes and output speed ratios each of which is produced by the combinations of two meshing pairs. We assume that there are standard radius of gears which take discrete values. We calculate each radius of gears so that standard gears are used as many as possible.

For example, suppose that we give the following specification for a four-speed gearbox shown in Figure 1. We denote a speed ratio by a combination of two meshing pairs $\langle g_i, g_j \rangle$ and $\langle g_k, g_l \rangle$ as $ratio(\langle g_i, g_j \rangle, \langle g_k, g_l \rangle)$. Then, we specify ratios as follows.

$$ratio(\langle g_1, g_3 \rangle, \langle g_5, g_7 \rangle) = 1 \quad \text{(a)}$$

$$ratio(\langle g_2, g_4 \rangle, \langle g_5, g_7 \rangle) = 2 \quad \text{(b)}$$

$$ratio(\langle g_1, g_3\rangle, \langle g_6, g_8\rangle) = 4 \quad \text{(c)}$$

$$ratio(\langle g_2, g_4\rangle, \langle g_6, g_8\rangle) = 8 \quad \text{(d)}$$

And we specify the sum of intervals as 10 and standard values of radius as $1, 2, 3, 4$.

Now, we modify this specification into a set of constraints expressed in equations. Let r_i be the radius of the gear g_i. Since $ratio(\langle g_i, g_j\rangle, \langle g_k, g_l\rangle) = p$ can be translated into an equation $r_i * r_k = r_j * r_l * p$, (a),...,(d) are translated into the following equations.

$$r_1 * r_5 = 1 * r_3 * r_7 \quad (1)$$

$$r_2 * r_5 = 2 * r_4 * r_7 \quad (2)$$

$$r_1 * r_6 = 4 * r_3 * r_8 \quad (3)$$

$$r_2 * r_6 = 8 * r_4 * r_8 \quad (4)$$

Note that one of the above four equations is redundant.

From the condition of meshing, the sum of radii between meshing pair must be equal to the interval between axes. We denote the interval between $Axis1$ and $Axis2$ as x_1 and the interval between $Axis2$ and $Axis3$ as x_2. Then, the following constraints exist.

$$r_1 + r_3 = x_1 \quad (5)$$

$$r_2 + r_4 = x_1 \quad (6)$$

$$r_5 + r_7 = x_2 \quad (7)$$

$$r_6 + r_8 = x_2 \quad (8)$$

From the specification of the sum of the intervals,

$$x_1 + x_2 = 10 \quad (9)$$

$(1), ..., (9)$ are hard constraints. To use standard gears as many as possible, we regard the possible standard values as soft constraints. In other words, we make the following soft constraints to be satisfied as many as possible for every radius r_i:

$$(r_i = 1) \vee (r_i = 2) \vee (r_i = 3) \vee (r_i = 4).$$

Now, we show how the above problem can be solved by using algebraic CHAL. In algebraic CHAL program, we can use the only constraint symbol, = and the algebraic function symbols such as +, *, and variables and fractions. The following program builds constraint hierarchy of the above problem.

```
gear :-
     ratio(1,2,4,8),distance(10),
     pos_val([r1,r2,r3,r4,r5,r6,r7,r8],[1,2,3,4]).
ratio(P1,P2,P3,P4) :-
    alg:r1*r5=P1*r3*r7, alg:r2*r5=P2*r4*r7,
    alg:r1*r6=P3*r3*r8, alg:r2*r6=P4*r4*r8.
distance(D) :-
    alg:x1+x2=D,
    alg:r1+r3=x1,alg:r2+r4=x1,
    alg:r5+r7=x2,alg:r6+r8=x2.
pos_val([],_).
pos_val([R|RL],VL) :- pos_val1(R,VL,C),chal:soft(C,0),pos_val(RL,VL).
pos_val1(R,[X],alg:R=X).
pos_val1(R,[X|Y],(alg:R=X;C)):- pos_val1(R,Y,C).
```

If we ask ?-gear, then algebraic CHAL firstly calculates a set of reduced constraints from hard constraints and then computes all simplified maximal consistent sets of constraints by solving constraint hierarchy. The result after considering soft constraints are shown in Figure 2. There are the two most preferable solutions. In both solutions, $r_1, ..., r_4$ are not standard gears. This is because hard constraints prevent those gears from being standard gears. Note that if constraints of possible standard values for gear were hard constraints, then we would not get any solution. In CHAL program, thanks to soft constraints, we can get solutions such that radii are standard values as many as possible. In this example, we can make $r_5, ..., r_8$ to be standard gears.

5 Conclusion

We compare our work with some related researches.

1. Borning et al. [2] were the first to propose the HCLP scheme. However, in [2], there is no logical formalization of the most preferable solutions. In this paper,

```
After considering soft constraints
solution
r1 = 8/3 .
r5 = 2 .
r3 = 4/3 .
r7 = 4 .
r2 = 16/5 .
r4 = 4/5 .
r6 = 4 .
r8 = 2 .
x1 = 4 .
x2 = 6 .

solution
r1 = 14/3 .
r5 = 1 .
r3 = 7/3 .
r7 = 2 .
r2 = 28/5 .
r4 = 7/5 .
r6 = 2 .
r8 = 1 .
x1 = 7 .
x2 = 3 .

solutionend

  12203msec
```

Figure 2: Solutions to Gear Design Problem

we provide a logical formalization by a variant of prioritized circumscription.

In [2], they discuss a relation of HCLP to nonmonotonic reasoning and claim that HCLP can handle the multiple extension problems of nonmonotonic logic. However, our result shows that a constraint hierarchy defined by HCLP is no more than a variant of prioritized circumscription. This means that HCLP can handle only multiple extension problems that can be solved by prioritized circumscription.

2. Baker et al. [1] give a theorem prover of prioritized circumscription. Since they use the finite domain closure axioms, they impose that their considered domain be finite.

   On the other hand, if we use algebraic CHAL, our domain is a complex number. So, semantic restriction does not always impose that the considered domain be finite.

Finally, we summarize the contributions of this paper.

1. We show a logical semantics of constraint hierarchy of HCLP by interpretation ordering.

2. From this semantics, we point out that a solution of constraint hierarchy can be regarded as the most preferable solution defined by semantically-restricted soft constraints. In the semantical restriction, the considered domain is fixed and only a logical combination of domain-dependent constraints can be used.

3. We propose a bottom-up algorithm of computing all maximal consistent constraint sets without any redundant calls of the constraint solver.

Acknowledgments I would like to thank Jun Arima from ICOT, Vladimir Lifschitz from University of Texas at Austin and Yuji Matsumoto from Kyoto University for instructive comments on this paper. Special thanks must go to Hiroyuki Sawada for tutoring me about a gear-box design.

Appendix A: An algorithm for solving constraint hierarchy

solve_constraint_hierarchy(CH, RRC)

% Solve constraint hierarchy CH with a set of reduced required constraints RRC.

begin

$PA := \{\langle \emptyset, RRC \rangle\}$

% PA is a set of pairs of ⟨Combined Constraints, Reduced Constraints⟩.

for every level L in CH from the strongest to the weakest **do**

begin

if $L \neq \emptyset$ **then**

begin

$NewPA := \emptyset$

for every pair $\langle Cs, RC \rangle$ in PA **do**

$NewPA$:=maximal_constraints($L, Cs, RC, NewPA$)

$PA := NewPA$

end

end

Take every RC of $\langle Cs, RC \rangle$ in PA to form a set, SC.

return SC

end (solve_constraint_hierarchy)

maximal_constraints(L, Cs, RC, PA)

% Find all maximal subsets in L which is consistent with RC.

begin

$QL := \{\langle Cs, \emptyset, RC, L \rangle\}$, $NGs := \emptyset$.

do

QL, NGs, PA :=maximal_constraints1(QL, NGs, PA)

until $QL = \emptyset$

return PA

end (maximal_constraints)

```
maximal_constraints1(QL, NGs, PA)
% Produce all extended consistent sets of constraints from QL.
% QL: a list of quadruple of the following sets of constraints:
%   (Combined Constraints, Used Constraints, Reduced Constraints, Rest)
% NGs: a set of contradictory combinations of constraints with RC.
begin
    NewQL := ∅
    for every element (Cs, UC, RC, Rest) in QL do
    begin
      while (Rest ≠ ∅) do
      begin
        Take one constraint C from Rest and delete C from Rest.
        % Note that Rest is decreased by one element for each while loop
        % so that every combination of constraints is checked only once.
        Add C to Cs to get NewCs
        % We extend Cs by adding C.
        if C is a disjunction then
          for every disjunct D in C do
            NewQL, NGs, PA :=
              maximal_constraints2(D, NewCs, UC, RC, Rest, NewQL, NGs, PA)
        else
          NewQL, NGs, PA :=
            maximal_constraints2(C, NewCs, UC, RC, Rest, NewQL, NGs, PA)
      end
    end
    return NewQL and NGs and PA
end (maximal_constraints1)
maximal_constraints2(C, NewCs, UC, RC, Rest, QL, NGs, PA)
% This is the main procedure of calculating maximal consistent sets of constraints.
begin
```

Add C to UC to get $NewUC$.

if there exists $NG \in NGs$ such that $NG \subseteq NewUC$ **then**

return QL and NGs and PA

% If we see that a subset of $NewUC$ is contradictory then

% we do not invoke *solve* and no longer extend $NewUC$.

$NewRC$:=solve(C, RC)

% If C and RC is consistent then

% solve(C, RC) returns a new set of reduced constraints

% otherwise it returns inconsistent information.

if $NewRC = inconsistent$ **then**

begin

Add $NewUC$ to NGs.

return QL and NGs and PA

% If we see that $NewRC$ is contradictory then we register it as nogoods

% and use it for further contradiction detecting and no longer extend $NewUC$.

end

if $Rest \neq \emptyset$ **then**

Add $\langle NewCs, NewUC, NewRC, Rest \rangle$ to QL

if there exists $\langle Cs', RC' \rangle \in PA$ such that $NewCs \subset Cs'$ **then**

return QL and NGs and PA

% If $NewCs$ is a strict subset of another combined constraints in PA

% then it is not a maximal consistent set.

Delete any $\langle Cs', RC' \rangle \in PA$ s.t. $Cs' \subset NewCs$.

% We delete every non-maximal consistent set from PA.

Add $\langle NewCs, NewRC \rangle$ to PA.

return QL and NGs and PA

end (maximal_constraints2)

References

[1] Baker, A. B. and Ginsberg, M. L.: *A Theorem Prover for Prioritized Circumscription, Proc. of IJCAI'89*, pp. 463 – 467 (1989).

[2] Borning, A., Maher, M., Martindale, A. and Wilson, M.: *Constraint Hierarchies and Logic Programming, Proc. of ICLP89*, pp. 149 – 164 (1989).

[3] Buchberger, B.: *Gröbner bases: An Algorithmic Method in Polynomial Ideal Theory*, In N. Bose, ed., *Multidimentional Systems Theory*, pp. 184 – 232, D. Reidel, Dordecht (1985).

[4] Descotte, Y. and Latombe, J.: *Making Compromises among Antagonist Constraints in a Planner, Artif. Intell.*, Vol. 27, pp. 183 – 217 (1985).

[5] Lifschitz, V.: *Computing Circumscription, Proc. of IJCAI85*, pp. 121–127 (1985)

[6] Lifschitz, V.: *On the Declarative Semantics of Logic Programs with Negation*, In J. Minker, ed., *Foundations of Deductive Databases and Logic Programming*, pp. 177 192, Morgan Kaufmann Publishers (1988).

[7] Sakai, K. and Aiba, A.: *CAL: A Theoretical Background of Constraint Logic Programming and its Applications, J. Symbolic Computation*, Vol. 8, pp. 589 – 603 (1989).

[8] Sakai, K. and Sato, Y.: *Application of Ideal Theory to Boolean Constraint Solving, Proc. of PRICAI90*, pp. 490 495.

[9] Satoh, K and Aiba, A.: *Hierarchical Constraint Logic Language: CHAL*, ICOT-TR 592 (1990).

[10] Satoh, K.: *Formalizing Nonmonotonic Reasoning by Preference Order, Proc. of InfoJapan90*, Part II, pp. 155 – 162 (1990).

[11] Satoh, K.: *Formalizing Soft Constraints by Interpretation Ordering, Proc. of ECAI90*, pp. 585 – 590 (1990).

[12] Smith, S. F., Fox, M. S. and Ow, P. S.: *Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems, AI Magazine*, Vol. 7, pp. 45 – 61 (Fall 1986).