Soft Consistencies for Weighted CSPs

Ken Brown

Cork Constraint Computation Centre, Department of Computer Science,
University College Cork, Ireland
k.brown@cs.ucc.ie

Abstract. A number of local consistency properties have recently been defined for weighted CSPs. We split the consistency definitions into a component arising from the constraint tuples, and a component arising from the unary weights, and develop the idea of weight consistencies. The components are then combined again to produce different levels of soft consistency.

1 Introduction

Soft CSPs associate weights with tuples of values ([1, 11]), and the aim of the solution process is to find a complete tuple with minimal total weight. Different methods of combining the individual weights give rise to different instantiations of the framework. Soft CSPs provide a convenient framework for constrained combinatorial optimisation problems. Recently, Schiex [10], Larrosa [7], Cooper [3] and Larrosa and Schiex [8] have proposed new definitions of local consistency for soft CSPs with a monotonic combination operator. Brown [2] proposed an algorithm for constraint problems with unary weights which maintains arc consistency interleaved with reasoning about bounds during search. In this paper, we compare and develop the different forms of local consistency. In particular, we start by separating out the components based on the constraint tuples and the components based on the weights. We can use this to define various local consistencies defined only on the weights and the constraint graph, by analogy with classical consistency definitions. We then combine the different components again, obtaining a series of local consistencies for weighted CSPs. We compare these to the definitions already proposed.

2 Background

A binary CSP is a triple $P = (X, D, C)$, where $X = \{1, ..., n\}$ is a set of variables, $D = \{D_1, ..., D_n\}$ is a set of domains of possible values for those variables, and C is a set of unary and binary constraints restricting the values variables can take simultaneously. A binary constraint between two variables i and j can be thought of as a function $C_{ij} : D_i \times D_j \rightarrow \{\bot, \top\}$, where a value of $\top$ indicates a pair is forbidden. Similarly, $C_i : D_i \rightarrow \{\bot, \top\}$. The scope of a constraint C_{ij} is $\{i, j\}$. A tuple of values assigned to variables is consistent if all constraints

whose scope is a subset of the variables in the tuple evaluate to $\bot$. A solution is a complete consistent tuple. The neighborhood of a variable i, $N_i = \{j : C_{ij} \in C\}$, is the set of all variables connected to i by a constraint.

Local consistency enforcing algorithms identify partial tuples which cannot be extended to complete solutions (i.e. locally inconsistent tuples). Various forms of local consistency have been proposed, including node consistency [9], arc consistency [9], k-inverse consistency [5], neighborhood inverse consistency [6], and singleton consistencies [4]. For comparing consistency properties, we use the notation $A \geq B$ (A stronger than B) if whenever property A holds in a problem then B holds, $A > B$ if there is at least one problem in which B holds but A doesn't, and $A \sim B$ (A incomparable to B) if neither $A \geq B$ nor $B \geq A$.

A *weighted* CSP (WCSP), following [7], extends the range of the constraints to $\{0, 1, ..., k\}$, where $0 = \bot$, and $k = \top$, and thus each constraint is a cost function. The aggregation operation for combining costs is defined by a valuation structure $S(k)$ where $k \in \{1, 2, ..., \infty\}$, $\oplus$ is such that $a \oplus b = min(k, a + b)$, and $\geq$ is the standard ordering on natural numbers. Note that $\oplus$ is monotonic (i.e. $a < b$ and $c < \top \Longrightarrow a \oplus c < b \oplus c$). X_t is the set of variables assigned in tuple t. If $B \subseteq X_t$, then the projection of t over B is denoted $t_{\downarrow B}$. The cost of a tuple t is defined to be

$$V(t) = \sum_{C_{ij} : \{i,j\} \subseteq X_t} C_{ij}(t_{\downarrow \{i,j\}}) \oplus \sum_{C_i : i \in X_t} C_i(t_{\downarrow \{i\}})$$

and the aim is to find a complete tuple with minimal cost (and cost $< \top$). Two WCSPs are defined to be equivalent if they have the same set of variables, and they have the same cost distribution over complete tuples. The *projection* of a constraint C_{ij} onto a value $a \in D_i$ transfers cost from C_{ij} to C_i. A projection adds $v \leq min_{b \in D_j} C_{ij}(a, b)$ to $C_i(a)$, and subtracts v from each $C_{ij}(a, b)$. The *extension* of a weight $C_i(a)$ over C_{ij} transfers cost from C_i to C_{ij}. An extension adds $v \leq C_i(a)$ to $C_{ij}(a, b)$ for each $b \in D_j$, and subtracts v from $C_i(a)$. Schiex [10] shows projection and extension preserve equivalence.

Various local consistency properties have recently been defined for WCSPs, and are defined below. A problem which cannot be made locally consistent by projection, extension or pruning values cannot have a complete solution with cost less than $\top$. In essence, these properties state that for each value there is at least one $\bot$-weighted tuple connecting it to a supporting value in a connected domain. There is a difference with respect to local consistency in CSPs, though: if a value in a CSP is arc-inconsistent, and the other domain is node consistent, then we can remove the value; in WCSP, we only remove a value if its weight $= \top$. Work must be done to project weights onto values in the hope that a weight of $\top$ is obtained. In what follows C_0 is a 0-arity constraint, and we assume similar definions of projection and extension for C_0. C_0 provides a lower bound on the cost of a solution.

Definition 1. *(Local consistencies for weighted CSPs) Variable i is:*

NC	*if* $\forall a \in D_i\ C_i(a) < \top$ *[7]*
AC	*if* *i is NC and* $\forall a \in D_i\ \forall j \in N_i\ \exists b \in D_j\ C_{ij}(a,b) = \bot$ *[10, 7]*
*NC**	*if* $\exists a_0 \in D_i\ C_i(a_0) = \bot$ *and* $\forall a \in D_i\ C_0 \oplus C_i(a) < \top$ *[7]*
*AC**	*if* *i is NC* and* $\forall a \in D_i\ \forall j \in N_i\ \exists b \in D_j\ C_{ij}(a,b) = \bot$ *[7]*
DAC	*if* $\forall a \in D_i\ \forall j (> i) \in N_i\ \exists b \in D_j\ C_{ij}(a,b) \oplus C_j(b) = \bot$ *[3]*
FDAC	*if* *i is AC and DAC [3]*
*DAC**	*if* *i is DAC and NC* [8]*
*FDAC**	*if* *i is DAC* and AC* [8]*

Consider the definition of AC. A value is arc consistent if it has a $\bot$-weighted support tuple for every relevant constraint, and its own weight is less than $\top$. This definition is a combination of two notions: (i) classical arc consistency, and (ii) what could be called weight consistency. However, the two components are at two different levels, since the weight consistency relates only to the node itself and not to its support across the arc. Brown [2] proposes an algorithm for solving CSPs with unary weights. The algorithm interleaves standard arc consistency maintenance with reasoning about weights induced by supporting values, and thus implicitly specifies another local consistency property. In this paper, we formalise and extend these ideas. We separate out the constraint components and weight components of the consistency definitions, extend the weight components, and then combine them again. We will show that new definitions of consistency can be obtained, stronger than the undirected consistencies above (NC, AC, NC*, AC*), but incomparable with the directed consistencies (DAC, FDAC, DAC*, FDAC*).

3 Weight Consistencies

First we define various local weight consistencies, focusing on the weights (unary constraints). For the binary constraints, we are only concerned with whether or not the constraint exists - we make no reference to the tuples. Each consistency property will disallow values which cannot be part of a complete tuple with cost less than the upper bound. *Node* weight consistency (NWC) will disallow values whose weight exceeds the bound. *Arc* weight consistency (AWC) will disallow values of a variable whose weight plus the minimum weight in a connected domain exceeds the bound. *Pair* weight consistency (pWC) will disallow values whose weight plus the minimum weight in any other domain exceeds the bound. *Neighborhood inverse* weight consistency (NIWC) will disallow values of a variable whose weight plus the sum of the minimum weights in the neighborhood exceeds the bound. *Full* weight consistency (FWC) will disallow values whose weight plus the sum of the minimum weights over all other domains exceeds the bound. Note that in a fully connected graph, AWC and NIWC become pWC and FWC respectively. Other weight consistencies are possible: for example *k-inverse* weight consistency is defined in the appendix.

Definition 2. *(Weight Consistencies) Variable i is:*
NWC *if* $\forall a \in D_i\ C_i(a) < \top$
AWC *if* $\forall a \in D_i\ \forall j \in N_i\ \exists b \in D_j\ C_i(a) \oplus C_j(b) < \top$
pWC *if* $\forall a \in D_i\ \forall j \neq i\ \exists b \in D_j\ C_i(a) \oplus C_j(b) < \top$
$NIWC$ *if* $\forall a \in D_i\ \exists (b_1, \ldots, b_t) \in \prod_{j \in N_i} D_j\ C_i(a) \oplus \sum_{j \in N_i} C_j(b_j) < \top$
FWC *if* $\forall a \in D_i\ \exists (b_1, \ldots, b_t) \in \prod_{j \neq i} D_j\ C_i(a) \oplus \sum_{j \neq i} C_j(b_j) < \top$
and for all weight consistencies above, P is xWC *if* $\forall i \in X$ *i is* xWC.

Theorem 1. *For any problem with at least one binary constraint*

a) $AWC > NWC$
b) $pWC > AWC$
c) $NIWC > AWC$
d) $NIWC \sim pWC$
e) $FWC > pWC$
f) $FWC > NIWC$

Proof. Figure 1 shows various inequalities. The dotted lines represent the constraint graph. Solid lines represent the supporting tuples. Tuples are assumed to have $\bot$ weight - if not, the weight will be attached. $\top$-weighted (i.e. forbidden) tuples are not drawn.

a) $\neq$: 1(i); $\geq$: if minimum weight in D_j is $\bot$, then for $a, b < \top, b > \bot, a < a \oplus b$
b) $\neq$: 1(ii); $\geq$: pWC considers all pairs, AWC only constrained pairs
c) $\neq$: 1(ii); $\geq$: AWC sums arcs separately, NIWC sums all arcs simultaneously
d) 1(iii) and 1(iv)
e) $\neq$: 1(iii); $\geq$: FWC sums weights over a complete tuple, pWC sums over pairs
f) $\neq$: 1(iv); $\geq$: FWC sums over all domains, NIWC over the neighborhood only
□

4 Combining Weight and Constraint Consistencies

We now combine the different weight consistencies with various constraint consistencies to get soft consistencies, and classify some of the existing soft consistencies. The first group of definitions are simple conjunctions of two definitions. If we integrate constraint consistency X with weight consistency Y, we will denote the resulting soft consistency by $s(X, Y)$. $s(AC, NWC)$ will disallow values with no support in a connected domain, or with a weight equal to the upper bound. $s(AC, AWC)$ will disallow values with no support in a connected domain, or whose weight plus the minimum weight from any connected domain is equal to the upper bound. $s(AC, FWC)$ will disallow values with no support in a connected domain or whose weight plus the sum of the minimum weights from all other domains is equal to the upper bound. Other definitions are possible, but are omitted.

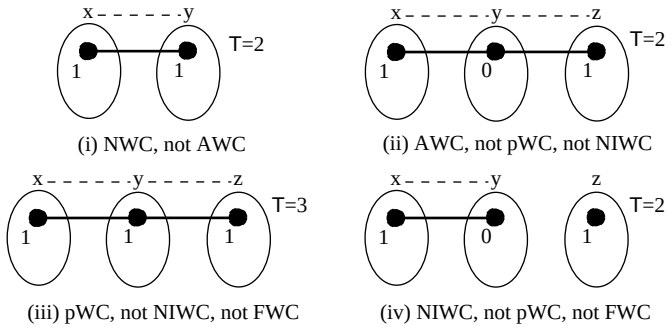

Fig. 1. weight consistency relationships

Definition 3. *Soft Consistencies*
Assuming there is no 0*-arity constraint, variable* i *is*
s(AC,NWC) if $\forall a \in D_i\ (\forall j \in N_i\ \exists b \in D_j\ C_{ij}(a,b) = \bot)$ *and* $C_i(a) < \top$
s(AC,AWC) if $\forall a \in D_i\ (\forall j \in N_i\ \exists b \in D_j\ C_{ij}(a,b) = \bot)$ *and*
$(\forall j \in N_i\ \exists b \in D_j\ C_i(a) \oplus C_j(b) < \top)$
s(AC,FWC) if $\forall a \in D_i\ (\forall j \in N_i\ \exists b \in D_j\ C_{ij}(a,b) = \bot)$ *and*
$(\exists (b_1, \ldots, b_t) \in \prod_{j \neq i} D_j\ \ C_i(a) \oplus \sum_{j \neq i} C_j(b_j) < \top)$

Note: $s(AC, NWC)$ is AC (Def 1).

Theorem 2. *For a WCSP P,* $s(AC, FWC)$ *and AC* are equivalent modulo* C_0

Proof. (i) P is AC* $\Rightarrow$ P can be made s(AC,FWC) by extending C_0 to any C_i
All values of all variables have a $\bot$-weighted support tuple for each constraint. Choose a variable k and maximally extend C_0 onto C_k, eliminating C_0 from the problem. Denote the new constraint C'_k. $\forall a \in D_k\ C'_k(a) = C_k(a) + C_0$. For each variable j, let v_j be $min_{b \in D_j}\{C_j(b)\}$. If $j \neq k\ C_j(v_j) = \bot$; $C'_k(v_k) = C_0$. Choose any value of any variable, say $a \in D_i$. Choosing the tuple of minimum weighted values, the last condition of $s(AC, FWC)$ is true, since

a) if $i \neq k$, $C_i(a) \oplus C'_k(v_k) = C_i(a) \oplus C_0 < \top$ by NC*
b) if $i = k$, $C'_k(a) = C_k(a) \oplus C_0 < \top$ by NC*.

(ii) P is $s(AC, FWC) \Rightarrow$ P can be made AC* by projecting all C_i onto C_0.
All values of all variables have a $\bot$-weighted support tuple for each constraint. For all variables, maximally project onto C_0. For each variable j, let v_j be $min_{b \in D_j}\{C_j(b)\}$. $C_0 = \sum_j C_j(v_j)$. Denote the resulting unary constraints C'_j for each j, and $C'_j(v_j) = \bot$. Choose the tuple of minimum values as before. The second condition of s(AC,FWC) then becomes $C_i(a) \oplus \sum_{j \neq i} C_j(v_j) < \top$. $C_i(a) = C'_i(a) \oplus C_i(v_i)$. So $C'_i(a) \oplus C_i(v_i) \oplus \sum_{j \neq i} C_j(v_j) = C'_i(a) \oplus C_0 < \top$, and hence P is NC* and so AC*. □

Theorem 3. *s(AC,FWC) > s(AC,AWC) > s(AC,NWC)*

Proof. by the same argument as for FWC > AWC > NWC □

The soft consistencies above are loosely coupled, and ignore the supports defined for arc consistency when computing the weight consistency. We can obtain stronger consistencies by requiring the same supports to be used in each component (where appropriate). If we strongly integrate constraint consistency X with weight consistency Y we will denote the resulting soft consistency by $S(X,Y)$. $S(AC,AWC)$ will disallow values whose weight added to the minimum support weight in a connected domain exceeds the bound. $S(AC,FWC)$ will disallow values which have no support in a domain or where the sum over the minimum support weights in connected domain and minimum value weights in unconnected domains exceeds the bound. This is the consistency property implicitly maintained by the algorithm in [2] for problems with unary weights. $S(NIC,FWC)$ will disallow values which cannot be extended to a consistent tuple in its neighborhood, or where the sum of its weight plus the weights of the values in the tuple plus the minimum value weights in unconnected domains exceeds the bound.

Definition 4. *(strong soft consistencies) Variable i is:*

S(AC,AWC) if $\forall a \in D_i\ \forall j \in N_i\ \exists b \in D_j$
$(C_{ij}(a,b) = \bot \text{ and } C_i(a) \oplus C_j(b) < \top)$

S(AC,FWC) if $\forall a \in D_i\ \exists (b_1,\ldots,b_t) \in \prod_{j \neq i} D_j$
$(j \in N_i \Rightarrow C_{ij}(a,b) = \bot \text{ and } C_i(a) \oplus \sum_{j \neq i} C_j(b_j) < \top)$

S(NIC,FWC) if $\forall a \in D_i\ \exists t = (a, b_1, \ldots, b_p) \in D_i \times \prod_{j \in N_i} D_j\ \exists (c_1,\ldots,c_q) \in \prod_{k \notin N_i} D_k$
$\forall C_{jk} \text{ s.t. } \{j,k\} \subseteq N_i \cup \{i\}\ C_{jk}(t_{\downarrow\{j,k\}}) = \bot \text{ and}$
$C_i(a) \oplus \sum_{j \in N_i} C_j(b_j) \oplus \sum_{k \notin N_i} C_k(c_k) < \top.$

Theorem 4. *(Strong soft consistency relationships)*

a) S(AC,AWC) > s(AC,AWC)
b) S(AC,FWC) > s(AC,FWC)
c) S(AC,FWC) > S(AC,AWC)
d) S(NIC,FWC) > S(AC,FWC)

Proof. The inequalities are shown in (a) Fig 2(i), (b) 2(i), (c) 2(ii) and (d) 2(iii).

a) ≥: S(AC,AWC) uses weight from the support, s(AC,AWC) from any value
b) ≥: as for (i)
c) ≥: S(AC,FWC) sums over all domains, S(AC,AWC) over constrained domains
d) ≥: S(NIC,FWC) requires a consistent tuple, S(AC,FWC) pairwise consistency.

□

Corollary 1. *S(AC,FWC) > AC* (modulo C_0)*

Proof. by Theorem 4(b) and Theorem 2. □

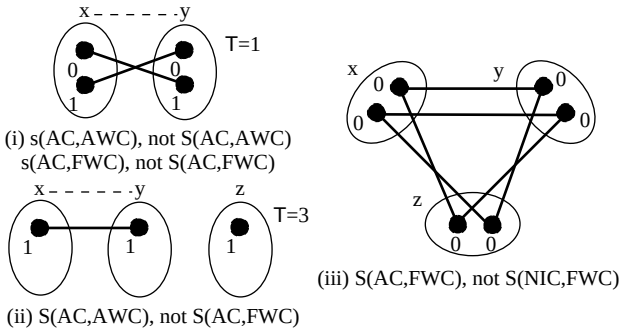

Fig. 2. strong soft consistency relationships

5 Star Node Consistency

Larrosa [7] introduced the 0-arity constraint C_0, into which all the unary constraints can be projected, and defined an enhanced node consistency NC* that required each domain to have a value with weight $\bot$. C_0 acts as a lower bound, collecting the minimum cost from each domain. We can apply this idea to weight consistencies, where we extend the definition to ensure that each variable is NWC*.

Definition 5. *Variable i is*

*NWC** *if* $\exists a \in D_i\ C_i(a) = \bot$ *and* $\forall a \in D_i\ C_0 \oplus C_i(a) < \top$

*AWC** *if* *i is NWC* and*
$\forall a \in D_i\ \forall j \in N_i\ \exists b \in D_j\ \ C_0 \oplus C_i(a) \oplus C_j(b) < \top$

*pWC** *if* *i is NWC* and*
$\forall a \in D_i\ \forall j \neq i\ \exists b \in D_j\ \ C_0 \oplus C_i(a) \oplus C_j(b) < \top$

*NIWC** *if* *i is NWC* and*
$\forall a \in D_i\ \exists (b_1, \ldots, b_t) \in \prod_{j \in N_i} D_j\ \ C_0 \oplus C_i(a) \oplus \sum_{j \in N_i} C_j(b_j) < \top$

*FWC** *if* *i is NWC* and*
$\forall a \in D_i\ \exists (b_1, \ldots, b_t) \in \prod_{j \neq i} D_j\ \ C_0 \oplus C_i(a) \oplus \sum_{j \neq i} C_j(b_j) < \top$

Theorem 5. *NWC* = AWC* = pWC* = NIWC* = FWC**

Proof. Every domain has a value with weight $\bot$. Therefore we can choose the b and b_j such that all the inequalities reduce to $C_0 \oplus C_i(a) < \top$, which must be true if each variable is NWC*. □

Since all the star weight consistencies are equivalent, when we loosely integrate them with standard consistencies, we get equivalent soft consistencies. However, we can still produce different strong soft consistencies. pWC* and

FWC* will not be considered, since they add no strength over AWC* and NIWC* respectively.

Definition 6. *(Star strong soft consistencies) Variable i is:*

S(AC,NWC)* *if* i *is NWC* and* $\forall a \in D_i\ \forall j \in N_i\ \exists b \in D_j\ C_{ij}(a,b) = \bot$

S(AC,AWC)* *if* i *is NWC* and* $\forall a \in D_i\ \forall j \in N_i\ \exists b \in D_j$
$(C_{ij}(a,b) = \bot$ *and* $C_0 \oplus C_i(a) \oplus C_j(b) < \top)$

S(AC,NIWC)* *if* i *is NWC* and* $\forall a \in D_i\ \exists (b_1, \ldots, b_t) \in \prod_{j \in N_i} D_j\ \forall j \in N_i$
$(C_{ij}(a,b) = \bot$ *and* $C_0 \oplus C_i(a) \oplus \sum_{j \in N_i} C_j(b_j) < \top)$

S(NIC,NIWC)* *if* i *is NWC* and* $\forall a \in D_i\ \exists (a, b_1, \ldots, b_t) \in D_i \times \prod_{j \in N_i} D_j$
$\forall C_{jk}$ *s.t.* $\{j,k\} \subseteq N_i \cup \{i\}$
$C_{jk}(t_{\downarrow\{j,k\}}) = \bot$ *and* $C_0 \oplus C_i(a) \oplus \sum_{j \in N_i} C_j(b_j) < \top$

Note: S(AC,NWC*) is AC* from [7].

Theorem 6. *(Star consistency relationships)*

a) S(AC,NWC) = s(AC,FWC)*
b) S(AC,AWC) > S(AC,NWC*)*
c) S(AC,NIWC) > S(AC,AWC*)*
d) S(AC,NIWC) = S(AC,FWC)*
e) S(NIC,NIWC) > S(AC,NIWC*)*

Proof. Inequalities for (b) and (c) are shown in Fig 3 (i) and (ii)

a) Proof of Theorem 3
b) $\geq$: S(AC,AWC*) sums over the weights on both sides of an arc, S(AC,NWC*) sums over weights on one side of an arc
c) $\geq$: S(AC,NIWC*) sums over all support weights, S(AC,AWC*) sums over pairs
d) by the same argument as (a), projecting over C_0
e) since NIC > AC

□

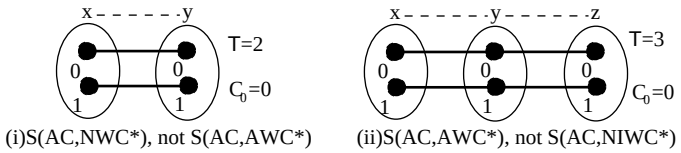

Fig. 3. star consistency relationships

6 Directed and Full Directed Consistency

Cooper [3] introduced DAC and FDAC, in which every value in a domain must have a $\perp$-weighted support in the domain in the direction of a directed arc. The aim of this is to move the weights across the arcs to the head, thus collecting higher weights and allowing for more value pruning. Larrosa and Schiex [8] then extended the definitions to DAC* and FDAC* (see Definition 1).

Theorem 7. *FDAC* > DAC*, *FDAC* > FDAC > DAC*

Proof. [3, 8] □

Theorem 8. *FDAC* ∼ S(AC,AWC*), DAC ∼ S(NIC,NIWC*)*

Proof. By Fig 4, and because FDAC* > DAC, and S(NIC,NIWC) > S(AC,AWC*) □

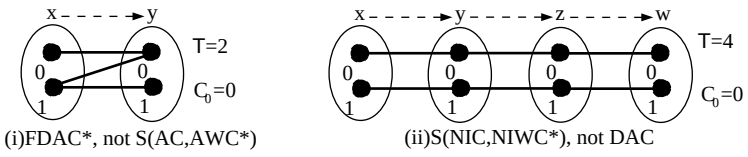

Fig. 4. incomparability with DAC and FDAC*

Both weight consistencies and directed consistencies gather information about costs across the constraint tuples. The directed consistencies do this by requiring that the weights are moved across directed arcs. For a good ordering, this is likely to result in higher unary weights on values at the start of the ordering, since those values are collecting the minimum cost for a complete tuple. The ordering is important; for example, Fig 4(a) would not be FDAC* if the ordering was reversed. The weight consistencies place a limit on costs in the whole connected neighborhood, and there is no ordering required, but will require the enforcing algorithm to maintain the neighborhood costs in separate data structures.

7 Conclusion and Future Work

We have shown that local consistency properties for weighted CSPs can be defined in terms of a constraint consistency component and a weight consistency component. We have defined a number of different levels of local consistency properties, and shown their relationship. We have also compared some of the existing consistency properties from the literature to this scheme. A diagram showing the main relationships is in Figure 5. Future work will focus on an algorithm for enforcing S(AC,NIWC*), and comparing to enforcing FDAC* [8].

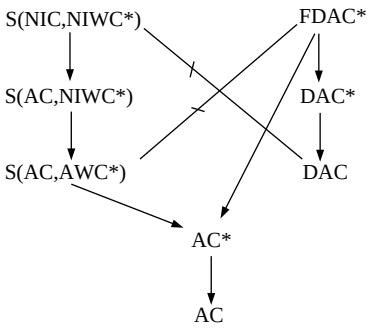

Fig. 5. soft consistency relationships

References

1. S. Bistarelli, U. Montanri, and F. Rossi. Constraint solving over semi-rings. In *Proc IJCAI'95*, pages 624–630, 1995.
2. K. Brown. Maintaining arc consistency in csps with unary weights. Technical report, Cork Constraint Computation Centre, TR-03-2003 2003.
3. M. C. Cooper. Reduction operations in fuzzy or valued constraint satisfaction. *Fuzzy Sets and Systems*, 134(3), 2003.
4. R. Debruyne and C. Bessiere. Some practicable filtering techniques for the constraint satisfaction problem. In *IJCAI'97*, pages 412–417, 1997.
5. E. G. Freuder. A sufficient condition for backtrack-bounded search. *Journal of the ACM*, 1985.
6. E. G. Freuder and C. Elfe. Neighborhood inverse consistency preprocessing. In *AAAI-96*, pages 202–208, 1996.
7. J. Larrosa. Node and arc consistency in weighted csp. In *AAAI-02*, pages 48–53, 2002.
8. J. Larrosa and T. Schiex. In the quest of the best form of local consistency for weighted csp. In *IJCAI'03*, 2003.
9. A. K. Mackworth. Consistency in netowrks of relations. *Artificial Intelligence*, 8:99–118, 1977.
10. T. Schiex. Arc consistency for soft constraints. In *CP2000*, pages 411–424, 2000.
11. T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *IJCAI'95*, pages 631–637, 1995.

A Appendix

k-inverse weight consistency disallows any value which cannot be extended to a weight consistent k-tuple.

Definition 7. *(k-inverse weight consistency)*
i is k-inverse weight consistent (kIWC) if

$\forall a \in D_i \ \forall tuples(i_1, \ldots, i_{k-1}) \ s.t. \ i_j \neq i \ \exists (b_1, \ldots, b_{k-1}) \in \prod_{j=1}^{k-1} D_{ij} \ C_i(a) \oplus \sum_{j=1}^{k-1} C_{ij}(b) < \top$

Theorem 9. *kIWC strengths*

a) $2IWC = pWC$
b) $k_1 > k_2 \Rightarrow k_1IWC > k_2IWC \ \forall k_1 \leq |X|$
c) $k < |X| \Rightarrow FWC > kIWC$, $k = |X| \Rightarrow FWC = kIWC$
d) $k < |X| \Rightarrow kIWC \sim NIWC$, $k = |X| \Rightarrow kIWC > NIWC$

Proof. Omitted □