Chapter 8

Advanced consistency methods: Relational consistency and Bucket Elimination

The definition of i-consistency is applicable to any constraint network, not just to binary ones. Frequently, it is more practical to restrict inference by the number of constraints participating in the inference rather than by the number of variables. Also, when most of the constraints' arity is higher than two, arc-consistency and path-consistency, as defined, rarely seem to be relevant.

Example 8.0.1 Consider the constraint network $\mathcal{R}$ over five integer domains where the constraints take the form of linear equations, and the domains are bounded by closed intervals:

$$\begin{array}{rl} D_x \in & [-2,3] \\ D_y \in & [-5,7] \\ R_{xyz}: & x+y=z \\ R_{ztf}: & z+t=f \end{array}$$

From the constraint R_{xyz} and the domain constraint D_x, we can infer the new constraint $R_{zy}: \;\; z-y \in [-2,3]$. From this new constraint and the domain D_y, we can also infer $D_z \in [-15,21]$. Although these inferences seem quite basic, they do not fit within our previous definitions of arc- and path- consistency. Since only a single constraint and a single domain are involved in each of the above inferences, they do have a flavor of basic arc-consistency. However, by definition, z is already arc-consistent relative to both x and y, since there is no binary restriction on the pair $\{x,z\}$ or on the pair $\{y,z\}$. Consider now the two constraints, $x+y=z$ and $z+t=f$. From these two constraints we can infer the new constraint $x + y + t = f$. Since the two constraints involve five different variables, this inference can be perceived as part of applying 5-consistency to the network. However, because it involves just two constraints, it seems to parallel path-consistency in a binary constraint network as well. □

The above examples demonstrate that our vocabulary for discussing local consistency does not conveniently accommodate straightforward inferences between constraints of arity greater than two. It would be natural indeed to view the first inference in Example 8.0.1. as arc-consistency, since only one constraint is involved, and the second as enforcing path-consistency (or 2-consistency), since only two constraints are involved.

8.1 Relational consistency

In this subsection we introduce the notions of *relational*-arc-consistency, *relational*-path-consistency, and, more generally, *relational-m*-consistency, all of which focus on the relations rather than on the variables as the basic entities.

We start with relational-arc-consistency. The question is, which property of arc-consistency do we choose to keep when extending to higher order constraints? That we deal with one constraint at a time? That we record domain constraints only? both? We will eventually express both, but we begin by focusing on the number of constraints involved. A constraint is *relational-arc-consistent* iff its projections over *any* subset of its scope are explicitly present in the network.

Definition 8.1.1 *(relational-arc-consistency) Let $\mathcal{R}$ be a constraint network over a set of variables $X = \{x_1, ..., x_n\}$, having domains $D_1, ..., D_n$, and let R_S be a relation (constraint) in $\mathcal{R}$ defined over a scope S. We say that R_S in $\mathcal{R}$ is* relational-arc-consistent *relative to a variable $x \in S$ iff any consistent instantiation of the variables in $S - \{x\}$ has an extension to a value in D_x that satisfies R_S. That is, if,*

$$\rho(S - \{x\}) \subseteq \pi_{S-\{x\}}(R_S \bowtie D_x) \tag{8.1}$$

(Recall that $\rho(A)$ is the set of all consistent instantiations of the subnetwork defined by A.) We say that a relation $R_{S'}$ where $S' = S - \{x\}$ is relational arc-consistent relative R_S iff for every tuple $t \in R_{S'}$ there exist a tuple $q \in R_S$ s.t., $q[S'] = t$. A relation R_S in $\mathcal{R}$ is relational-arc-consistent *if it is relational-arc-consistent relative to every variable in S. A network is relational-arc-consistent iff every relation is relational-arc-consistent.*

Naturally, if relational-arc-consistency is not satisfied, it can be imposed by adding a constraint, as dictated by Eq. (8.1). Namely, we can add the constraint $R_{S-\{x\}}$,

$$R_{S-\{x\}} \leftarrow \pi_{S-\{x\}}(R_S \bowtie D_x) \tag{8.2}$$

If we want to ensure that a single constraint only is defined over each scope even in the case where arc-consistency already exists, then we can use instead the rule that tighten the current constraint:

$$R_{S-\{x\}} \leftarrow R_{S-\{x\}} \cap \pi_{S-\{x\}}(R_S \bowtie D_x). \tag{8.3}$$

Including such constraints ensures the condition in Eq. (8.1) since determining the consistency of a partial solution on A will include testing the added constraint. The following two examples demonstrate the concept of relational arc-consistency.

Example 8.1.2 Consider a network with one constraint $R_{xyz} = \{(a,a,a),(a,b,c),(b,b,c)\}$. This relation is not relational arc-consistent, but if we add the projection $R_{xy} = \{(a,a),(a,b),(b,b)\}$, then R_{xyz} will become relational arc-consistent relative to $\{z\}$. To make this network relational-arc-consistent, we would have to add all the projections of R_{xyz} with respect to all subsets of its variables. □

Example 8.1.3 Consider again Example 8.0.1. We already noticed that the network is arc-consistent relative to the *variable-based* definition. However, it is not relational-arc-consistent; there is a consistent assignment to z and y that cannot be extended to a value in x's domain while satisfying the constraint $x + y = z$. If we add the constraints $R_{zy}: \ z - y \in [-2, 3]$, $R_{zx}: \ z - x \in [-7, 10]$, $D_z: \ z \in [-7, 10]$, and $R_{ft}: \ f - t \in [-7, 10]$, then the resulting network, $\mathcal{R}'$, augmented with the above four new constraints, is relational-arc-consistent. □

We next define *relational-path-consistency.* The definition involves the simultaneous consistency of two constraints, disregarding their arity.

Definition 8.1.4 (Relational path-consistency) *Let $\mathcal{R}$ be a constraint network over a set of variables $X = \{x_1, ..., x_n\}$ having domains $D_1, ..., D_n$, and let R_S and R_T be two constraints in $\mathcal{R}$. We say that R_S and R_T are* relational-path-consistent *relative to a variable $x \in S \cap T$ iff any consistent instantiation of the variables in $S \cup T - \{x\}$ has an extension to a value in the domain of x, D_x, that satisfies R_S and R_T simultaneously; that is, iff*

$$\rho(A) \subseteq \pi_A(R_S \bowtie R_T \bowtie D_x) \tag{8.4}$$

where $A = (S \cup T) - \{x\}$. A pair of relations R_S and R_T is relational-path-consistent *iff it is relational-path-consistent relative to every variable in $S \cap T$. A network is relational-path-consistent iff every pair of relations is relational-path-consistent.*

Note that the definition of relational-path-consistency subsumes relational-arc-consistency if we allow non-distinct pairs of relations.

Example 8.1.5 We saw that the network in Example 8.0.1 is not relational-path-consistent since we can assign x, y, f, and t values that are consistent relative to the relational-arc-consistent network $\mathcal{R}'$ generated in Example 8.1.3. For example, the assignment $(< x, 2 >, < y, -5 >, < t, 3 >, < f, 15 >)$ is consistent since only domain restrictions are applicable, but there is no value of z that simultaneously satisfies $x + y = z$ and $z + t = f$. To make the two constraints relational path-consistency relative to z we should deduce the constraint $x + y + t = f$ and add it to the network. □

Relational-consistency can be extended to m relations in a similar way:

Definition 8.1.6 (Relational-m-consistency) *Let $\mathcal{R}$ be a constraint network over a set of variables $X = \{x_1, ..., x_n\}$, and let $R_{S_1}, \ldots, R_{S_m}$ be m distinct relations in $\mathcal{R}$, where $S_i \subseteq X$. We say that $R_{S_1}, \ldots, R_{S_m}$ are* relational-m-consistent *relative to $x \in \bigcap_{i=1}^{m} S_i$ iff any consistent instantiation of the variables in $A = \bigcup_{i=1}^{m} S_i - \{x\}$ has an extension to x that satisfies $R_{S_1}, \ldots, R_{S_m}$ simultaneously; that is, iff*

$$\rho(A) \subseteq \pi_A(\bowtie_{i=1}^{m} R_{S_i} \bowtie D_x) \tag{8.5}$$

where $A = \bigcup_{i=1}^{m} S_i - \{x\}$. A set of relations $\{R_{S_1}, \ldots, R_{S_m}\}$ is relational-m-consistent *iff it is* relational-m-consistent *relative to every variable in $\bigcap_{i=1}^{m} S_i$. A network is relational-m-consistent iff every set of m relations is relational-m-consistent. A network is strongly relational-m-consistent if it is relational-i-consistent for every $i \leq m$.*

Relational-arc- and path-consistency correspond to relational-1- and relational-2-consistency, respectively.

Example 8.1.7 Consider the constraint network over the set of variables $\{x_1, x_2, x_3, x_4, x_5\}$, where each variable has the domains $D_i = \{$a,b,c$\}$, $1 \leq i \leq 5$, and the relations are given by

$R_{2,3,4,5}$ = { (a,a,a,a), (b,a,a,a), (a,b,a,a), (a,a,b,a), (a,a,a,b) },
$R_{1,2,5}$ = { (b,a,b), (c,b,c), (b,a,c) }

The constraints are not relational-arc-consistent. For example, the instantiation ($x_2 = a$, $x_3 = b$, $x_4 = b$) is a consistent instantiation as it satisfies all the applicable constraints (trivially so, as there are no constraints defined strictly over $\{x_2, x_3, x_4\}$ or over any of their subsets), but it does not have an extension to x_5 that satisfies $R_{2,3,4,5}$. Similarly, the constraints are not relational-path-consistent. For example, the instantiation ($x_1 = c$, $x_2 = b$, $x_3 = a$, $x_4 = a$) is consistent (again, trivially so), but it does not have an extension to x_5 that satisfies $R_{2,3,4,5}$ and $R_{1,2,5}$ simultaneously. If we add the unary constraints $R_2 =$

$R_3 = R_4 = \{\text{a}\}$ and $R_1 = R_5 = \{\text{b}\}$ we get an equivalent network which is both relational-arc- and relational-path-consistent. The reason is that since all the variables' domains have a singleton value, the set of solutions over every subset of variables will contain a single tuple only, the one that is extended to a full solution. □

When all the constraints are binary, relational-m-consistency is identical (except for minor preprocessing) to variable-based i-consistency where $i = m + 1$; otherwise, the conditions are different. The definition of relational m-consistency is similar (but not identical to) that of variable-based m-consistency if applied to the dual view of the problem. In this view, the constraints are perceived to be variables, their allowed tuples are their respective domains, and two such constraint-variables are constrained by binary constraints if they have variables in common.

8.1.1 Space-bound vs time-bounds

Up to now, whether we used variable-based or relational-based consistency, if a subnetwork of size i was used as the basis of inference, a constraint having scope $i-1$ was inferred. This dependency between the size of the consulted subnetwork and the arity of the inferred constraint can be avoided. We next discuss this option of separating the size of the network we consult from the size of the resulting inferred constraint. In *relational* (i,m)-*consistency* defined below, m always indexes the cardinality of a set of relations, and i corresponds to the arity of the constraints that are tested for local consistency and the size of the recorded constraint.

Definition 8.1.8 (relational (i,m)-consistency) *A set of relations* $\{R_{S_1}, \ldots, R_{S_m}\}$ *is* relationally (i,m)-consistent *iff for every subset of variables* A of size i, *any consistent assignment to* A *can be extended to an assignment to* $\bigcup_{i=1}^m S_i - A$ *that satisfies* $R_{S_1},\ldots,R_{S_m}$, *simultaneously. A network is* relationally (i,m)-consistent *iff every set of* m *relations is relationally* (i,m)*-consistent. A network is* strong relational (i,m)-consistent *iff it is relational* (j,m)*-consistent for every* $j \le i$.

Figure 8.1 presents algorithm Relational-Consistency, $RC_{(i,m)}$, a brute-force algorithm for enforcing strong relational-(i,m)-consistency on a network $\mathcal{R}$. It parallels the brute-force algorithms such as *AC-1* and *PC-1*. Algorithms for enforcing relational consistency that are not scope restricted will be referred to as RC_m, and they can be obtained by using size $i = |\cup_{i=1}^m S_i| - 1$ in step 3 of the algorithm.

Note that R_A stands for the current unique constraint specified over the scope A. If no constraint exists, then R_A is the universal relation over A. The algorithm takes any m relations that may or may not be relational-(i,m)-consistent and enforces relational

RELATIONAL-CONSISTENCY$(\mathcal{R}, i, m)$ $(RC_{(i,m)})$
input: a network $\mathcal{R} = (X, D, C)$
output: a network $\mathcal{R}'$ relational-(i,m)-consistent, equivalent to $\mathcal{R}$.
1. **repeat**
2. $\quad Q \leftarrow \mathcal{R}$
3. $\quad$ **for** every m relations $R_{S_1}, \ldots, R_{S_m} \in Q$
$\quad$ and every subset A of size i, $A \subseteq \bigcup_{i=1}^{m} S_i$
4. $\quad\quad R_A \leftarrow R_A \cap \ \pi_A(\bowtie_{i=1}^{m} R_{S_i})$
5. $\quad\quad$ **if** R_A is the empty relation
6. $\quad\quad\quad$ **then** exit and return the empty network
7. $\quad\quad$ **endif**
8. $\quad$ **endfor**
10. **until** $Q = \mathcal{R}$

Figure 8.1: Relational-(i,m)-consistency

m-consistency by tightening the relation among subsets of variables of size i. We call the operation in Step 4 *extended composition* (extending also the revise operator) since it generalizes the composition operation defined on binary relations.

Definition 8.1.9 (extended composition) *The extended composition of relation R_{S_1}, ..., R_{S_m} relative to a subset of variables $A \subseteq \bigcup_{i=1}^{m} S_i$, denoted $EC_A(R_{S_1}, \ldots, R_{S_m})$, is defined by*

$$EC_A(R_{S_1}, \ldots, R_{S_m}) = \pi_A(\bowtie_{i=1}^{m} R_{S_i})$$

When the operator is applied to m relations, it is called extended m-composition. *If the projection operation is restricted to subsets of size i, it is called* extended (i,m)-composition.

Algorithm $RC_{(i,m)}$ computes the closure of $\mathcal{R}$ with respect to extended (i,m)-composition which is clearly computationally expensive for large i and m, though it can be improved in a manner parallel to the improvements of path-consistency algorithms.

The two most practical relational-consistency algorithms are those which are based on single constraints at the most and which record constraints on just a subset of the variables, within existing scopes. Two such algorithms are $RC_{(1,1)}$, also called *domain-propagation*, and the more general relational arc-consistency, RC_1. Domain propagation derives domain restrictions based on a single constraint. It can be defined by the rule:

$$D_x \leftarrow \pi_x(R_S \bowtie D_x) \tag{8.6}$$

where $x \in S$. Relational arc-consistency can be derived by a rule recording for every constraint R_S, a constraint on $S - \{x\}$ for every $x \in S$. Namely:

$$R_{S-\{x\}} \leftarrow \pi_{S-\{x\}}(R_S \bowtie D_x) \tag{8.7}$$

Both rules consider all assignments that agree with a single constraint and a single domain variable. While the first rule records (projects over) the single variable, the second rule projects on all the variables in the constraint scope excluding x. The first shrinks the domains of x more explicit, while the second makes the constraint on $S - \{x\}$ more explicit. Notice that for binary constraints, these two rules are identical. Specializing $RC_{(i,m)}$ for the two rules in EQ. 8.6 and EQ. 8.7 yields algorithms which we call DOMAIN-PROPAGATION ($RC_{(1,1)}$) and RELATIONAL ARC-CONSISTENCY (RC_1), respectively.

The complexity of algorithm DOMAIN-PROPAGATION is clearly polynomial. The complexity of RELATIONAL-ARC-CONSISTENCY is polynomial (in the input) as well, but unless all the constraints have bounded scopes, the algorithm may be exponential in the largest scope size.

8.2 Directional-relational-consistency

The class of *DIC*, variable-based directional consistency algorithms introduced in Chapter 4, can be extended to the relational consistency case in a natural way.

Definition 8.2.1 (directional relational consistency) *Given a network $\mathcal{R}$ and an ordering $d = (x_1, \ldots, x_n)$, $\mathcal{R}$ is m-*directionally relationally consistent *iff for every subset of constraints $\{R_{S_1}, \ldots, R_{S_m}\}$ whose latest variable in any of the constraints' scopes is x_l, then for every A $A \subseteq \{x_1, \ldots, x_{l-1}\}$, every consistent assignment to A can be extended to x_l while satisfying all the relevant constraints in $\{R_{S_1}, \ldots, R_{S_m}\}$ simultaneously.*

Figure 8.2 presents algorithm DIRECTIONAL-RELATIONAL-CONSISTENCY, DRC_m. The algorithm, which also uses the bucket data-structure, places in the bucket of x_j all the relations whose latest variable is x_j. Subsequently, buckets are processed in descending order from last to first, where in each bucket the extended m-composition operator (step 6 of DRC_m), is applied to every subset of m relations. In addition to extended composition (step 6 of DRC_m), the algorithm may include two optional steps of *simplification* and *instantiation*. The simplification step ensures that each bucket contains relations defined on distinct subsets of variables that are not subsumed by each other. The instantiation

DIRECTIONAL-RELATIONAL-CONSISTENCY (DRC_m)
Input: a constraint network $\mathcal{R} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$, an ordering $d = (\ x_1, \ldots, x_n)$
output: An m-directional-consistent network, along d
1. Partition the constraints into, $bucket_1, \ldots, bucket_n$, where $bucket_i$ contains all constraints whose highest variable is x_i.
2. **for** $p \leftarrow n$ **downto** 1 **do**
3. **simplification**: **for** every $S_i, S_j \in bucket_p$, such that $S_i \supseteq S_j$ **do**
   $R_{S_i} \leftarrow \pi_{S_i}(R_{S_i} \bowtie R_{S_j})$
4. **instantiation: if** $bucket_p$ contains the constraint $x_p = u$ **then**
   for every $R_{S_i} \in bucket_p$ **do**
   $R_{S_i - \{x_p\}} \leftarrow \pi_{S_i - \{x_p\}}(\sigma_{x_p = u} R_{S_i})$
   if not empty, add the result to its appropriate bucket, else exit.
5. **else** (the general case)
   $j \leftarrow \min\{\text{cardinality of bucket}_p,\ m\}$
6. **for** every j relations $R_{S_1}, \ldots, R_{S_j}$ in $bucket_p$ **do**
   $A \leftarrow \cup S_i - \{x_p\}$, $R_A \leftarrow \pi_A(\bowtie_i S_i)$
7. **if** R_A is not empty **then** add R_A to its appropriate bucket
8. **else** exit and return the empty network
9. **return** $E^m(R) = \bigcup_{j=1}^{n} bucket_j$

Figure 8.2: Algorithm DRC_m

step is invoked when a bucket contains a value assignment, and in that case, instead of applying extended composition, each relation in the bucket is assigned the given value, and the resulting function is moved into its appropriate bucket. These two steps should also be included in any practical implementation of algorithm Adaptive-Consistency.

Example 8.2.2 We use a crossword puzzle (see Figure 8.3) to illustrate algorithm DRC_2, and Adaptive-Consistency as well, as the two algorithms coincide on this example. As we have seen in Chapter 2, one possible formulation of the problem is to have a variable for each square that can hold a character, $x_1, \ldots, x_{13}$; the domains of the variables are the alphabet letters; and the constraints are the possible words. For this example, the constraints are given by,

$R_{1,2,3,4,5}$ = {(H,O,S,E,S), (L,A,S,E,R), (S,H,E,E,T), (S,N,A,I,L), (S,T,E,E,R)}
$R_{3,6,9,12}$ = {(A,L,S,O), (E,A,R,N),(H,I,K,E), (I,R,O,N), (S,A,M,E)}

$$\begin{aligned} R_{5,7,11} &= \{(\text{E,A,T}),\ (\text{L,E,T}),\ (\text{R,U,N}),\ (\text{S,U,N}),\ (\text{T,E,N}),\ (\text{Y,E,S})\} \\ R_{8,9,10,11} &= R_{3,6,9,12} \\ R_{10,13} &= \{(\text{B,E}),\ (\text{I,T}),\ (\text{N,O}),\ (\text{U,S})\} \\ R_{12,13} &= R_{10,13}. \end{aligned}$$

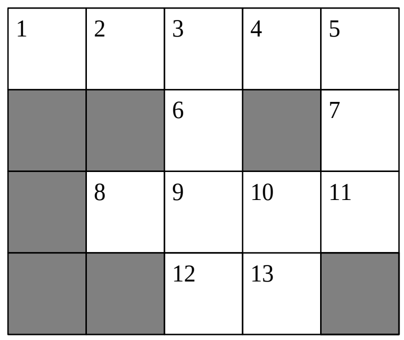


Figure 8.3: A crossword puzzle

Let us perform several iterations of DRC_2, where the ordering of the variables is $d = (x_{13}, x_{12}, \ldots, x_1)$. Thus, x_1 is the last variable in the ordering and x_{13} is the earliest. The derivation is described schematically in Figure 8.4. As a shorthand, we will call the bucket for each x_i by the name bucket$_i$. We denote the original relations by $R's$ and the new relations (created during the processing of DRC_2) by H's. Our first step is to distribute each original R relation into the bucket of the lowest-indexed variable in its scope (that is, into the bucket of the variable that comes latest in the ordering). Next, we process each of the buckets, starting at the end of the ordering. The bucket for x_1 contains the single relation $R_{1,2,3,4,5}$. Processing bucket$_1$ generates the relation,

$$\begin{aligned} H_{2,3,4,5} &= \pi_{2,3,4,5}(R_{1,2,3,4,5}) \\ &= \{(\text{O,S,E,S}),\ (\text{A,S,E,R}),\ (\text{H,E,E,T}),\ (\text{N,A,I,L}),\ (\text{T,E,E,R})\}, \end{aligned}$$

and places it in the bucket of variable x_2, which is processed next. The bucket for x_2 contains only this single relation $H_{2,3,4,5}$. Processing bucket$_2$ adds the relation,

$$H_{3,4,5} = \pi_{3,4,5}(H_{2,3,4,5}) = \{(\text{S,E,S}),\ (\text{S,E,R}),\ (\text{E,E,T}),\ (\text{A,I,L}),\ (\text{E,E,R})\},$$

which it adds to the bucket of variable x_3. This bucket is processed next and we see that it contains the relations $R_{3,6,9,12}$ and $H_{3,4,5}$. Processing bucket$_3$ adds one relation,

$$\begin{aligned} H_{4,5,6,9,12} &= \pi_{4,5,6,9,12}(H_{3,4,5} \bowtie R_{3,6,9,12}) \\ &= \{(\text{E,S,A,M,E}),\ (\text{E,R,A,M,E}),\ (\text{E,T,A,R,N}),\ (\text{I,L,L,S,O}),\ (\text{E,R,A,R,N})\} \end{aligned}$$

to the previously empty bucket of variable x_4. Processing $bucket_4$ adds the relation,

$H_{5,6,9,12}$ = {(S,A,M,E), (R,A,M,E), (T,A,R,N), (L,L,S,O), (R,A,R,N)}.

The bucket for x_5 now contains the relations $R_{5,7,11}$ and $H_{5,6,9,12}$. Processing bucket$_5$ adds the relation,

$H_{6,7,9,11,12}$ ={(A,U,M,N,E), (A,E,R,N,N), (L,E,S,T,O), (A,U,R,N,N) }.

The bucket for x_6 contains only this newly generated relation $H_{6,7,9,11,12}$. Processing bucket$_6$ adds the relation,

$H_{7,9,11,12}$ = {(U,M,N,E), (E,R,N,N), (E,S,T,O), (U,R,N,N),},

to the bucket for x_7. Processing bucket$_7$ adds the relation,

$H_{9,11,12}$ = {(M,N,E), (R,N,N), (S,T,O)},

to the bucket of x_9. Next we process the bucket of x_8, containing only the original relation $R_{8,9,10,11}$, and when processed it adds the relation,

$H_{9,10,11}$ = {(L,S,O), (A,R,N), (I,K,E), (R,O,N), (A,M,E)}.

The bucket for x_9 now contains the relations $H_{9,10,11}$ and $H_{9,11,12}$. Processing bucket$_9$ adds the relation,

$H_{10,11,12}$ = {(O,N,N)}.

The bucket for x_{10} contains the relations $R_{10,13}$ and $H_{10,11,12}$. Processing bucket$_{10}$ adds the empty relation. Since the empty relation was derived, the algorithm stops and reports that the network is inconsistent. □

8.2.1 Complexity

The complexity of DRC_m, is not likely to be polynomial even for $m = 2$ since, as we will see, it solves NP-complete problems. Nevertheless, as observed for variable-based consistency enforcing, its worst-case complexity can be bounded using the *induced width* of the constraint graph. For convenience we redisplay the primal and dual constraint graphs of the crossword puzzle of Figure 8.3 in Figure 8.5.

We observe that the induced-width of the primal graph along the processing ordering $d = x_1, ..., x_{13}$, is 5. As observed for DIC$_i$, enforcing directional relational consistency transforms its primal constraint graph into the induced ordered graph. Since the complexity of recording a new constraint is exponential in the scope of these constraints, and since the scope of new constraints is identical to the induced width of a processed variable, we can bound the complexity of directional-relational-consistency using the induced width:

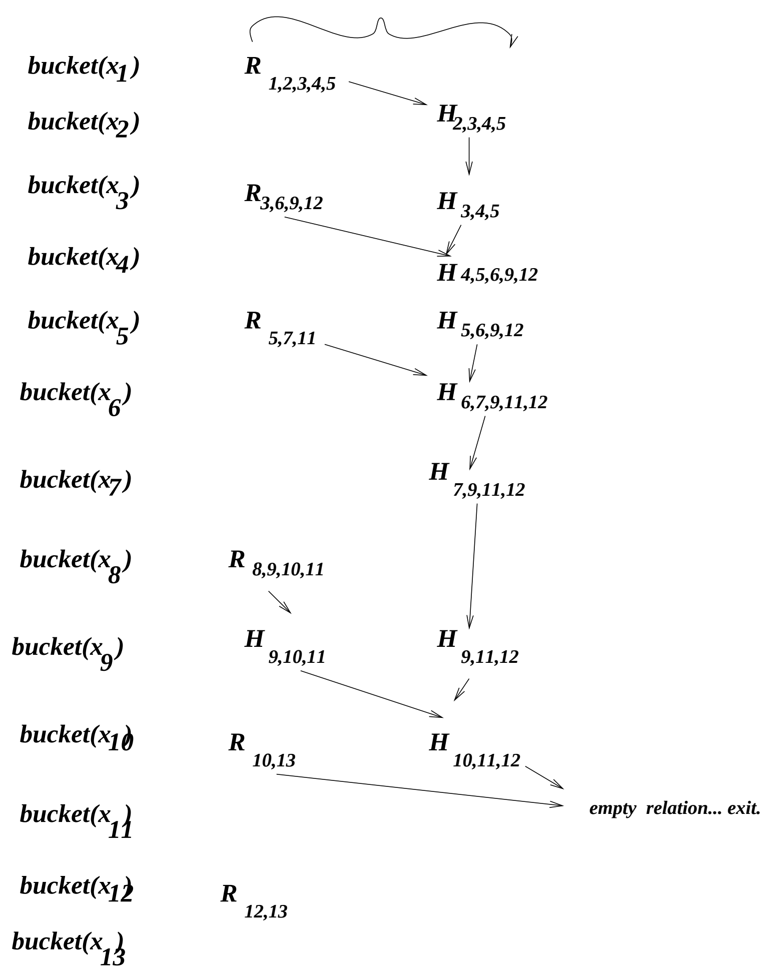


Figure 8.4: Example of applying DRC_2 to the crossword puzzle

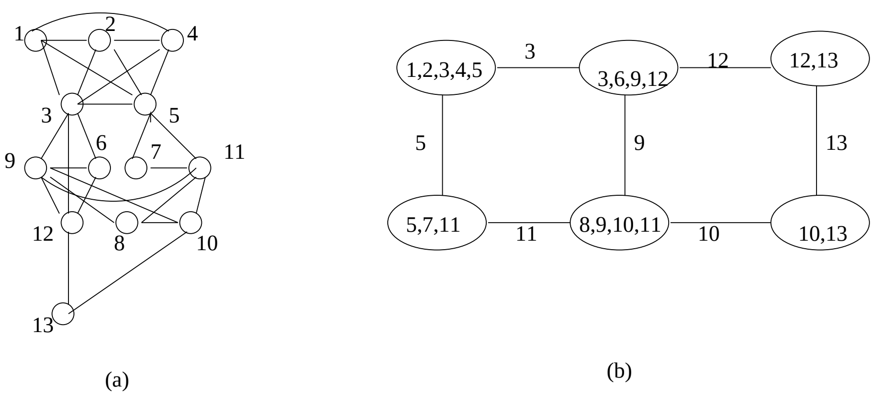


Figure 8.5: Primal and dual constraint graphs of the crossword CSP.

Theorem 8.2.3 (complexity of directional relational consistency) *Given a network $\mathcal{R}$, the time complexity and size of the network generated by DRC_m along ordering d is $O(nm \cdot (2^m k^2)^{(w^*(d)+1)})$, where $w^*(d)$ is the induced with of the network's constraint graph along ordering d. In particular, the time complexity of DRC_2 is $O(n(4k^2)^{(w^*(d)+1)})$.*

Proof: Since the number of variables mentioned in any bucket is at most $w^*(d)+1$, the number of relations in a bucket is bounded by $O(2^{w^*(d)+1})$, and the number of subsets of size m is $O(2^{(w^*(d)+1)m})$. Also, the number of tuples in each relation is bounded by $k^{w^*(d)+1}$. The complexity of an m-way join of relations of size $k^{w^*(d)+1}$ can be bounded by $O(m \cdot k^{2(w^*(d)+1)})$ since the size of the relation resulting from every pair-wise join is still bounded by $k^{w^*(d)+1}$ and thus, m consecutive joins do not multiply but only add. Consequently, the overall complexity is $O((nm) \cdot 2^{(w^*(d)+1)m} \cdot k^{2(w^*(d)+1)})$ as stated. $\Box$

The only case for which DRC_m is guaranteed to be polynomial occurs when $m = 1$.

The case of DRC_1 and DRC_2

Lemma 8.2.4 *The complexity of DRC_1 is $O(n \cdot e^2 \cdot t^2)$ when e is the number of input relations, and t bounds the number of tuples in each relation.*

Proof: We have n buckets to process. Each bucket will not contain more than e relations, at any time. The reason is that extended 1-composition involves projections and intersections only, which add only a linear number of constraints and which takes $O(t \cdot e)$ steps. Simplification of a bucket takes $O(e^2 \cdot t^2)$, yielding the result. $\Box$

Interestingly, whenever we apply adaptive-consistency to the crossword puzzle, at most two relations will ever reside in a single bucket, no matter what variable ordering is selected (see Example 8.2.2 and Figure 8.4). In other words, for the crossword puzzles, adaptive-consistency coincides with DRC_2.

Proposition 8.2.5 *1. Crossword puzzles can be made directional backtrack-free by DRC_2,*
2. The complexity of DRC_2 applied to crossword puzzles is $O(n \cdot k^{2(w^*(d)+1)})$.

Proof: Let $\mathcal{R}$ be a crossword puzzle instance. We will show that after processing by adaptive-consistency, each bucket have at most two relations. Therefore, adaptive-consistency coincides with DRC_2. Let us annotate each variable in a constraint by a "+″ if it appears in a horizontal word and by a "−″ if it appears in a vertical word. In the initial specification each variable appears in at most two constraints and each annotated variable appears in just one constraint (with that annotation). We show that this property is maintained throughout the algorithm's performance. The annotation of the variables in the constraint resulting from extended 2-composition inherits the annotation in the parent constraints. If a variable appeared with annotation "+" in one, and annotation "−" in the other, its annotation in the resulting constraint will be "+,−". The claim can be proved by induction on the processed buckets. Assume that after processing buckets $x_n, \ldots, x_i$ *all* the constraints appearing in the union of all the buckets from $bucket_{i-1}$ to $bucket_1$, satisfy that each annotated variable appears in at most one constraint. When processing $bucket_{i-1}$, since it contains only two constraints (otherwise 2 constraints will contain multiple annotations of variable x_{i-1}), it generates a single new constraint. Assume that the constraint is added to the bucket of x_j. Clearly, if x_j is annotated positively (respectively negatively) in the added constraint, $bucket_j$ cannot contain already a constraint with a positive (respectively, negative) annotation of x_j. Otherwise, it means that before processing bucket $i-1$, there were two constraints with positive (respectively negative) annotation of x_j, one in the bucket of x_{i-1} and one in the bucket of x_j, which contradicts the induction hypothesis. A very similar argument can be applied to the multiple annotation case. $\Box$

8.3 Domain and constraint tightness

Interestingly, RC_2 and DRC_2 can generate backtrack-free representations, for a variety of problems. Two such classes are Boolean formulas and linear algebraic inequalities over discrete integer domains. We next discuss relevant properties that relates domain and constraint tightness to a level of local consistency that ensures global consistency.

8.3.1 Domain tightness

The first property shows that constraint networks having smaller domain size are easier.

Theorem 8.3.1 *A constraint network $\mathcal{R} = (X, D, C)$ over domains with 2 values or less which is relational 2-consistent is globally consistent.*

Proof: Let $\vec{a}_{i-1} = (a_1, ..., a_{i-1})$ be a consistent assignment to variables $x_1, ...x_{i-1}$. We want to show that there is a value $< x_i, a_i >$ that will be consistent with $\vec{a}_{i-1}$. Since there are two possible values for a_i lets call them "0" and "1". Lets define by A_0 all the *relevant* constraints (those whose scope contains x_i and a subset of $x_1, ...x_{i-1}$) that violated by $(\vec{a}_{i-1}, x_i = 0)$ and by A_1 the corresponding set of relevant constraints violated by $(\vec{a}_{i-1}, x_i = 1)$. Clearly, A_0 and A_1 are not empty, or else we will have a consistent extension to x_i. Lets pick $R_1 \in A_0$ and $R_2 \in A_1$ (R_1 and R_2 may be equal). Therefore R_1 and R_2 are at most two constraints that are cannot consistently extend a common consistent partial assignment (the projection of $\vec{a}_{i-1}$ on the scopes of R_1 and R_2 excluding x_i) to x_i in their common scope, contradicting strong relational 2-consistency. $\Box$.

Theorem 8.3.1 implies that if we enforce relational 2-consistency we will make any bi-valued network globally consistent. The argument can be easily extended to directional relational consistency, namely a bi-valued network that is directional relational 2-consistent is directionally globally consistent. In other words, algorithm DRC_2 is complete for bi-valued theories. Remember that we already observed this property for binary constraints. However, while bi-valued binary constraints are tractable, non-binary bi-valued networks are not because variable-based path-consistency can be enforced in polynomial time while relational 2-consistency cannot.

The reader may guess that Theorem 8.3.1 can be extended to larger domains. Indeed, if the domain size of a constraint network has at most k values, strong relational k-consistency ensures global consistency. Consequently DRC_k globally solves any k-domain constraint problem such as the k-graph coloring problem. We will leave the proof of the general case to the reader.

Theorem 8.3.2 (domain tightness and local consistency) *A constraint network $\mathcal{R} = (X, D, C)$ over domains with at most k values which is relational k-consistent is globally consistent.* $\Box$

8.3.2 Constraint tightness

The notion of domain tightness can be extended from unary relations (domains) to arbitrary relations having any scope size, allowing a similar relationship with bounded local consistency.

Definition 8.3.3 (m-tight) *A constraint relation R_S of arity r is called* m-tight *if, for any variable $x_i \in S$ and any instantiation $\bar{a}$ of the remaining $k-1$ in $S - \{x_k\}$, either there are at most m extensions of $\bar{a}$ to x_i that satisfy R_S, or there are exactly $|D_i|$ such extensions.*

Notice that given a constraint network with c constraints each with arity at most r and each with at most t tuples in the relation, determining the least m such that all c of the constraints are m-tight requires $O(c \cdot r \cdot t)$ time.

Example 8.3.4 We illustrate the definitions using the following network $\mathcal{R}$ with variables $\{x_1, x_2, x_3, x_4\}$, domains $D_i = \{$a,b,c$\}$, $1 \leq i \leq 4$, and constraints,

$$\begin{aligned} R_{S_1} = \{&(a,a,a),\ (a,a,c),\ (a,b,c),\ (a,c,b),\ (b,a,c),\\ &(b,b,b),\ (b,c,a),\ (c,a,b),\ (c,b,a),\ (c,c,c)\},\\ R_{S_2} = \{&(a,b),\ (b,a),\ (b,c),\ (c,a),\ (c,c)\},\\ R_{S_3} = \{&(a,b),\ (a,c),\ (b,b),\ (c,a),\ (c,b)\},\end{aligned}$$

where $S_1 = \{x_1, x_2, x_3\}$, $S_2 = \{x_2, x_4\}$, and $S_3 = \{x_3, x_4\}$.

It can be verified that all of the constraints are 2-tight. As a partial verification of the binary constraint R_{S_3}, consider the extensions to variable x_3 given instantiations of the variable x_4. For the instantiation $< x_4, a >$ there is 1 extension to x_3; for $< x_4, b >$ there are 3 extensions (but $|D_3| = 3$ so the definition of 2-tightness is still satisfied); and for $< x_4, c >$ there is 1 extension. □

We next introduce and prove the main relationship:

Theorem 8.3.5 (constraint-tightness and local consistency) *If a constraint network $\mathcal{R} = (X, D, C)$ with m-tight constraints is strongly relationally $(m+1)$-consistent, then it is globally consistent.*

Proof: Let $R_{S_1}, \ldots, R_{S_l}$ be l relations that constrain a variable x, let k be the size of the domain of variable x, and let $\bar{a}$ be an instantiation of all of the variables except for x that are constrained by the l relations (i.e., $\bar{a}$ is an instantiation of the variables in $(S_1 \cup \cdots \cup S_l) - x)$. We will show that if each relation is m-tight, for some $1 \leq m < k$, and if for every subset of $m+1$ or fewer relations from $\{R_{S_1}, \ldots, R_{S_l}\}$, there exists at least one extension of $\bar{a}$ to x that satisfies each of the relations in the subset (which is the strong relational (m+1)-consistency) then there exists at least one extension of $\bar{a}$ to x that satisfies all l relations. This will concludes the proof of the theorem.

Let $a_1, a_2, \ldots, a_d$ be the k elements in the domain of x. Assume to the contrary that an extension of $\bar{a}$ to x that satisfies all of the l constraints does not exist. Then, for each element a_i in the domain of x there must exist at least one constraint that does not allow a_i. Let c_i denote one of the constraints that does not allow a_i. By construction, the set $c = \{c_1, c_2, \ldots, c_k\}$ is a set of constraints for which there does not exist a consistent extension of $\bar{a}$ to x that satisfies each of the constraints in the set. We will show that for every possible value of m, this leads to a contradiction as follows.

Case 1 ($m = k-1$): The contradiction is immediate as $c = \{c_1, c_2, \ldots, c_k\}$ is a set of m+1 constraints for which there does not exist an extension to x that satisfies every constraint in the set, contradicting relational (m+1)-consistency.

Case 2 ($m = d-2$): The nominal size of the set $c = \{c_1, c_2, \ldots, c_k\}$ is $m+2$. We claim, however, that there is a repetition in c and that the true size of the set is $m+1$. Assume to the contrary that $c_i \neq c_j$ for $i \neq j$. Recall c_i is a constraint that does not allow a_i, $i = 1, \ldots, k$ and consider $\{c_1, c_2, \ldots, c_{k-1}\}$. This is a set of $m+1$ relations so relational (m+1)-consistency implies that there must exist an a_i of x such that every constraint in the set allows. The only possibility is a_k of x. Now consider $\{c_1, \ldots, c_{k-2}, c_k\}$. Again, this is a set of $m+1$ relations so there must exist an a_i that every relation in the set allows. This time the only possibility is a_{k-1}. Continuing in this manner, we can show that relation c_1 must allow $a_k, a_{k-1}, \ldots, a_2$; that is, c_1 must allow exactly $m+1$ extensions. This contradicts the condition of m-tightness of c_1. Therefore, it must be the case that $c_i = c_j$ for some $i \neq j$. Thus, the set c is of size $m+1$ and therefore contradicts relational (m+1)-consistency.

Case 3 ($m = d-3$), ..., **Case d-1** ($m = 1$): The remaining cases are similar. In each case we argue that (i) there are repetitions in the set $c = \{c_1, c_2, \ldots, c_k\}$, (ii) the true size of the set c is $m+1$, and (iii) a contradiction is derived by appealing to strong relational (m+1)-consistency. Thus, there exists at least one extension of $\bar{a}$ to x that satisfies all of the relations. $\square$

Notice that Theorem 8.3.5 always specifies a level of strong consistency that is less than or equal to the level of strong consistency required by domain-tightness in Theorem 8.3.2. The level of required consistency is equal only when $m = k-1$ and is less when $m < k-1$. For example, the graph k-colorability problem provides examples of networks where domain-tightness and constraint-tightness give the same bound on the sufficient level of local consistency since the constraints are $(k-1)$-tight.

Notice also that once we enforce some level of relational consistency the tightness level may increase or decrease. The parameter m will increase if a new constraint is being recorded on a subset of variables that were not explicitly constained before, and the new constraint has a largert m-tightness value.

Another important special case are bi-valued constraint networks. For example, the satisfiability of propositional cnfs provide an example of networks with domains of size two. Relations which constrain variables with domains of size two are 1-tight and any additional constraints that are added to the network as a result of enforcing strong relational 2-consistency will also be 1-tight. This is another example where domain-tightness and constraint-tightness yields the same required consistency level.

8.4 Inference for Boolean theories

We saw in the previous section that for Boolean constraints enforcing relational 2-consistency is sufficient. Namely extended-2-composition is the operator relevant to consistency enforcing for such theories. It turns out that for *cnf* theories the extended 2-composition is equivalent to pair-wise resolution. Both these observation yields well known processing algorithms for the task of propositional satisfiability.

8.4.1 Consistency for Boolean propositional *cnf* theories

To review, let's denote propositional variables by uppercase letters $P, Q, R, \ldots$, propositional literals (i.e., $P, \neg P$) by lowercase letters $p, q, r, \ldots$, and disjunctions of literals, or *clauses*, by $\alpha, \beta, \ldots$. For instance, $\alpha = (P \vee Q \vee R)$ is a clause. A *unit clause* is a clause of size 1. The notation $(\alpha \vee T)$, when $\alpha = (P \vee Q \vee R)$ is shorthand for the disjunction $(P \vee Q \vee R \vee T)$, and $\alpha \vee \beta$ denotes the clause whose literals appears in either α or β. The *resolution* operation over two clauses $(\alpha \vee Q)$ and $(\beta \vee \neg Q)$ results in a clause $(\alpha \vee \beta)$, thus eliminating Q. *Unit resolution* is a resolution operation when one of the clauses is a unit clause. A formula φ in conjunctive normal form (*cnf*) is a set of clauses $\varphi = \{\alpha_1, \ldots, \alpha_t\}$ that denotes their conjunction. The set of *models* of a formula φ, denoted $models(\varphi)$, is the set of all satisfying truth assignments to all its propositional variables. A clause α is *entailed* by φ, $\varphi \models \alpha$, if and only if α is true in all models of φ. A clause α is a *prime implicate* of φ if and only if $\varphi \models \alpha$ and $\not\exists \beta \subseteq \alpha$ such that $\varphi \models \beta$. A Horn formula is a *cnf* formula whose clauses all have at most one positive literal. A clause is positive if it contains only positive literals and is negative if it contains negative literals only. A k-*cnf* formula is one whose clauses are all of length k or less.

Example 8.4.1 Consider the following *cnf* formula:

$$\varphi = \{(F \vee X \vee Y \vee \neg Z) \wedge (F \vee Z) \wedge (X \vee Y \vee F)\}.$$

Formula φ can be viewed as a constraint network where each clause corresponds to a constraint defined by its models. It is easy to see that the clauses are relationally arc-consistent since their projection on any subset of propositional symbols results in the universal relations allowing everything (no unary constraints are available). The first two clauses are also relationally path-consistent relative to F since any truth assignment to X, Y, Z can be extended by assigning "true" to F, thereby satisfying both clauses. Similarly, the two clauses are relationally path-consistent relative to Z since any consistent assignment to X, Y, F has to satisfy the third clause of φ and therefore by assigning $r = true$ the first two clauses are satisfied. The reader is invited to check that the entire set of clauses is relationally path-consistent. □

Resolution

We next show that resolution is equivalent to extended 2-composition.

Example 8.4.2 Consider the two clauses $\alpha = (P \vee \neg Q \vee \neg O)$ and $\beta = (Q \vee \neg W)$. Now let the relation $R_{PQO} = \{000, 100, 010, 001, 110, 101, 111\}$ be the models of α and the relation $R_{QW} = \{00, 10, 11\}$ be the models of β. Resolving these two clauses over Q generates the resolvent clause $\gamma = res(\alpha, \beta) = (P \vee \neg O \vee \neg W)$. The models of γ are: $\{(000, 100, 010, 001, 110, 101, 111\}$. It is easy to see that $EC_{PQW}(R_{PQO}, R_{QW})$ yields the models of γ. □

Indeed,

Lemma 8.4.3 *The* resolution *operation over two clauses,* $(\alpha \vee Q)$ *and* $(\beta \vee \neg Q)$*, results in a clause* $(\alpha \vee \beta)$ *satisfying:* $models(\alpha \vee \beta) = EC_{Q'}(models(\alpha \vee Q), models(\beta \vee \neg Q))$*, where* Q' *is the union of scopes of both clauses excluding* Q*.* □

Specializing algorithm RC_2 (Figure 8.1) to cnf theories, replacing extended 2-composition with resolution yields an algorithm that applies pair-wise resolution to the theory until no new resolvents can be generated.

Since as shown, constraint networks with domain size 2 that are relational 2-consistent are globally consistent (Theorem 8.3.1), the resolution algorithm, enforcing relational 2-consistency, generates a globally consistent and therefore backtrack-free representation. When the resolution algorithm terminates, and if we also eliminate redundant clauses by subsumption elimination (namely if $\alpha \subseteq \beta$ we can delete β) we will get all the *prime implicates* of the cnf theory. We can therefore conclude that the set of all prime implicates of a theory is globally consistent.

As noted in Chapter 2, the structure of a propositional theory can be described by its *interaction graph*. The interaction graph of a propositional theory φ, denoted $G(\varphi)$, is an undirected graph that contains a node for each propositional variable and an edge for each pair of nodes that correspond to variables appearing in the same clause. For example, the interaction graph of theory $\varphi_1 = \{(\neg C), (A \vee B \vee C), (\neg A \vee B \vee E), (\neg B \vee C \vee D)\}$ is shown in Figure 8.6a.

Notice that the interaction graph of a theory processed by resolution should be augmented with new edges reflecting the added resolvents. For example, resolution over variable A in φ_1 generates a new clause $(B \vee C \vee E)$, so the graph of the resulting theory has an edge between nodes E and C as shown in Figure 8.6b.

Unit resolution

Next observe that since a domain constraint in cnfs theories appears as a unit literal, the relational-arc consistency rule (EQ. 8.7) translates to unit-resolution. Indeed, UNIT

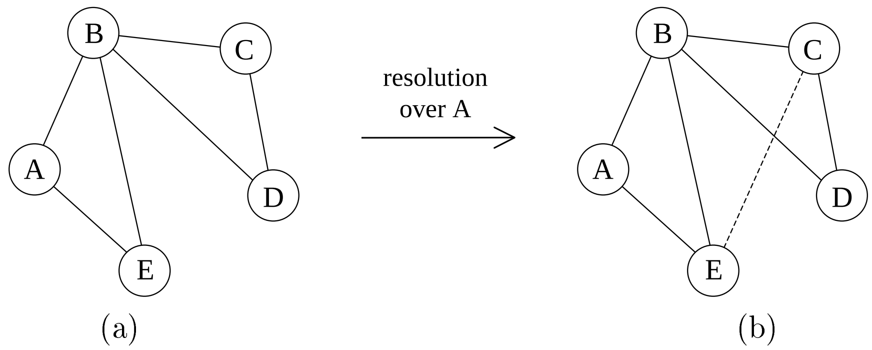


Figure 8.6: (a) The interaction graph of theory $\varphi_1 = \{(\neg C), (A \vee B \vee C), (\neg A \vee B \vee E), (\neg B \vee C \vee D)\}$, and (b) the effect of resolution over A on that graph.

PROPAGATION, an algorithm equivalent to relational arc-consistency, is the most useful inference algorithm for *cnf* theories. Unlike general relational arc-consistency it can be implemented in linear time. Algorithm UNIT-PROPAGATION is given in Figure 8.7.

Theorem 8.4.4 *Algorithm* UNIT-PROPAGATION *has a linear time complexity.*

Proof: Each unit-resolution or unit-subsumption operation causes either the elimination of a clause or the shortening of a clause by one. Therefore the number of unit-resolutions is at most the length of the *cnf* formula. □.

8.4.2 Directional resolution (DR)

Looking at *directional* relational 2-consistency for *cnf* theories and specializing DRC_2 for *cnf*s by replacing extended 2-composition with resolution and the instantiation step by unit resolution, results in algorithm DIRECTIONAL-RESOLUTION (DR).

Algorithm DIRECTIONAL RESOLUTION (DR) is presented in Figure 8.8. We call its output theory, $E_d(\varphi)$, the *directional extension* of φ. Given an ordering $d = (Q_1, ..., Q_n)$, all the clauses containing Q_i that do not contain any symbol higher in the ordering are placed in the bucket of Q_i, denoted $bucket_i$. The algorithm processes the buckets in the reverse order of d. Processing of $bucket_i$ means resolving over Q_i all the possible pairs of clauses in the bucket and inserting the resolvents into appropriate lower buckets.

Note that if the bucket contains a unit clause (Q_i or $\neg Q_i$), only unit resolutions are performed.

Example 8.4.5 Given the input theory $\varphi_1 = \{(\neg C), (A \vee B \vee C), (\neg A \vee B \vee E), (\neg B \vee C \vee D)\}$, and an ordering $o = (E, D, C, B, A)$, the theory is partitioned into buckets and processed by directional resolution in reverse order. Resolving over variable A produces a

Procedure UNIT-PROPAGATION
Input: A cnf theory, φ, $d = Q_1, ..., Q_n$.
Output: An equivalent theory such that every unit clause does not appear in any non-unit clause.

queue = all unit clauses.
while queue is not empty, do.
    $T \leftarrow$ next unit clause from Queue.
    for every clause β containing T or $\neg T$
        if β contains T delete β (subsumption elimination)
        else, For each clause $\gamma = resolve(\beta, T)$.
        if γ, the resolvent, is empty, the theory is unsatisfiable.
        else, add the resolvent γ to the theory and delete β.
        if γ is a unit clause, add to Queue.
    endfor.
endwhile.

Figure 8.7: Algorithm unit propagation

DIRECTIONAL-RESOLUTION

Input: A *CNF* theory φ, an ordering $d = Q_1, \ldots, Q_n$ of its variables.

Output: A decision of whether φ is satisfiable. If it is, a theory $E_d(\varphi)$, equivalent to φ, else an empty directional extension.

1. **Initialize:** generate an ordered partition of clauses into buckets. $bucket_1$, ..., $bucket_n$, where $bucket_i$ contains all clauses whose highest literal is Q_i.
2. **for** $i \leftarrow n$ **downto** 1 process $bucket_i$:
3.   **if** there is a unit clause **then** (the instantiation step)
           apply unit-resolution in $bucket_i$ and place the resolvents in their right buckets.
           if the empty clause was generated, theory is not satisfiable.
4.   **else** resolve each pair $\{(\alpha \vee Q_i), (\beta \vee \neg Q_i)\} \subseteq bucket_i$.
           if $\gamma = \alpha \vee \beta$ is empty, return $E_d(\varphi) = \{\}$, theory is not satisfiable
           else determine the index of γ and add it to the appropriate bucket.
5. **return** $E_d(\varphi) \leftarrow \bigcup_i bucket_i$

Figure 8.8: Directional-resolution

new clause $(B \vee C \vee E)$, which is placed in $bucket_B$. Resolving over B then produces clause $(C \vee D \vee E)$ which is placed in $bucket_C$. Finally, resolving over C produces clause $(D \vee E)$ which is placed in $bucket_D$. Directional resolution now terminates, since no resolution can be performed in $bucket_D$ and $bucket_E$. The output is a non-empty directional extension $E_o(\varphi_1)$. Once the directional extension is available, model generation begins. There are no clauses in the bucket of E, the first variable in the ordering, and therefore E can be assigned any value (e.g., $E = 0$). Given $E = 0$, the clause $(D \vee E)$ in $bucket_D$ implies $D = 1$, clause $\neg C$ in $bucket_C$ implies $C = 0$, and clause $(B \vee C \vee E)$ in $bucket_B$, together with the current assignments to C and E, implies $B = 1$. Finally, A can be assigned any value since both clauses in its bucket are satisfied by previous assignments. The initial partitioning into buckets along the ordering $d = (E, D, C, B, A)$ as well as the buckets' contents generated by the algorithm following resolution over each bucket are depicted in Figure 8.9(a). □

As implied by Theorem 8.3.1, DRC_2 is guaranteed to generate a backtrack-free representation along the order of processing. Indeed, as already observed in the above example, once all the buckets are processed, and if the empty clause was not generated, a truth assignment (a model), can be assembled in a backtrack-free manner by consulting $E_d(\varphi)$ using the order d.

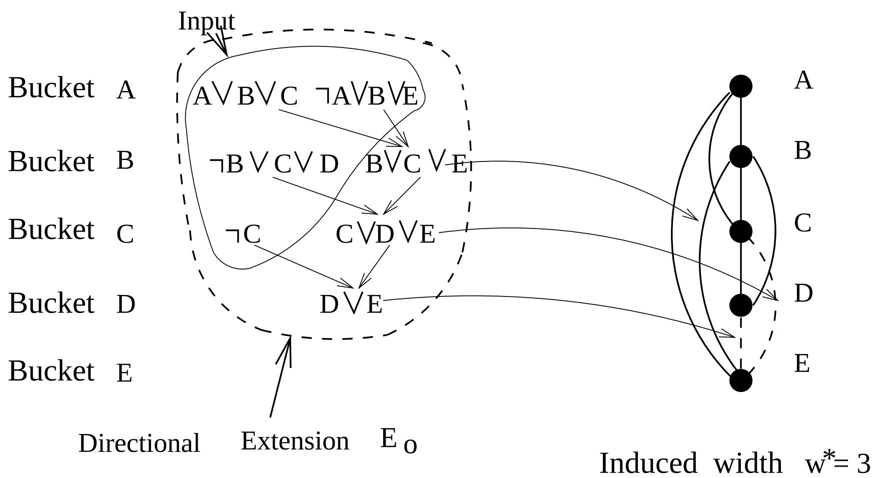


Figure 8.9: A schematic execution of directional resolution using ordering $d = (E, D, C, B, A)$

Not surprisingly, the complexity of directional-resolution is exponentially bounded (time and space) in the *induced width* of the theory's interaction graph along the order of processing. We can prove this based on the complexity of DRC_2 (yielding $O(n16^{w*+1})$). Notice that the graph of theory φ along the ordering d depicted in Figure 8.9(b) has induced width of 3. Simple direct arguments yield a somewhat tighter bound.

Lemma 8.4.6 *Given a theory φ and an ordering $o = (Q_1, ..., Q_n)$, if Q_i has at most k parents in the induced graph along o, then the bucket of Q_i in $E_o(\varphi)$ contains no more than 3^{k+1} clauses.*

Proof: Given a clause α in the bucket of Q_i, there are three possibilities for each parent P of Q_i: either P appears in α, or $\neg P$ appears in α, or neither of them appears in α. Since Q_i also appears in α, either positively or negatively, there are no more than $2 \cdot 3^k < 3^{k+1}$ different clauses in the bucket. □ ■

Since the number of parents of each variable is bounded by the induced-width along the order of processing we get,

Theorem 8.4.7 *(complexity of DR)*
Given a theory φ and an ordering of its variables o, the time complexity of algorithm DR along o is $O(n \cdot 9^{w^*_o})$, and $E_o(\varphi)$ contains at most $n \cdot 3^{w^*_o+1}$ clauses, where w^*_o is the induced width of φ's interaction graph along o. □

8.4.3 Tractable Boolean theories

2-SAT

Note that algorithm directional-resolution is tractable for 2-cnf theories, because 2-cnfs are closed under resolution (the resolvents are of size 2 or less) and because the overall number of clauses of size 2 is bounded by $O(n^2)$, yielding $O(n \cdot n^2) = O(n^3)$ complexity. In this case, unordered resolution is also tractable. Exploiting w^* we can get the following:

Theorem 8.4.8 *Given a 2-cnf theory φ, its directional extension $E_o(\varphi)$ along any ordering o is of size $O(n \cdot {w^*}^2)$, and can be generated in $O(n \cdot {w^*}^2)$ time.*

Obviously, directional-resolution is not the best algorithm for solving 2-SAT problem. It is known that 2-SAT can be solved in linear time [143]. Note, however, that directional-resolution not only decides consistency but also compiles the theory into one that can produce each model in time linear in the output theory.

Horn theories

Horn theories are cnf formulas where each clause has at most one positive literal. They can be written as a collection of implications also called *Horn rules* of the form,

$$P \leftarrow Q_1 \wedge Q_2 \wedge ... \wedge Q_r$$

where all the Q_i's and P are positive atoms. P is called the head of the rule and $\{Q_1, ..., Q_r\}$ is its antecedent. Some of the rules may have empty heads.

Interestingly, the consistency of Horn theory can be fully determined by unit-propagation. Assume we apply unit-propagation to a Horn theory. If the algorithm terminates without generating an empty clause, its outputs can be partition into a set of unit clauses and a disjoint set of non-unit clauses. Consequently, if we assign each unit clause its truth value and each of the rest of the propositions a negative truth value, the resulting assignment does not violate any clause, because each clause must contain a negative literal. In summary,

Theorem 8.4.9 *The consistency of Horn theories can be determined by unit propagation. If the empty clause is not generated the theory is satisfiable.* $\Box$

Since unit-propagation is merely relational-arc consistency we just showed that relational arc-consistency is complete for Horn satisfiability. In Chapter 11 we will show that Horn theories as well as 2-SAT theories can be extended to tractable classes over multi-valued variables in general CSPs.

8.5 Row convex constraints

In this section we identify a property of constraints called *row convexity* and show its usefulness in deciding when relational path consistency is sufficient to guarantee global consistency. Testing for the row convexity property involves determining whether there exists an ordering of the domains of the variables such that all of the constraints are simultaneously row convex. The property of row-convexity is orthogonal to tightness. Its main impact is likely to be when the domain and constraints are not tight.

Row convexity extends some special binary constraint types such as functional and monotone constraints. For simplicity sake we present definitions and analyze the case of binary constraints first.

Definition 8.5.1 (functional) *A binary relation R_{ij} expressed as a (0,1)-matrix is* functional *if and only if there is at most one one in each row and in each column of R_{ij}.*

Definition 8.5.2 (monotone) *Given some ordering of the domain of values for all variables, a binary relation R_{ij} expressed as a (0,1)-matrix is* monotone *if the following conditions hold: if $(a,) \in R_{ij}$ and $c \geq a$, then $(b, c) \in R_{ij}$, and if $(a,) \in R_{ij}$ and $c \leq b$, then $(a, c) \in R_{ij}$.*

Definition 8.5.3 (row convex) *A binary relation R_{ij} represented as a (0,1)-matrix is* row convex *if in each row (column) all of the ones are consecutive; that is, no two ones within a single row are separated by a zero in that same row (column).*

For example, consider the simple constraint network with variables x_1 and x_2 and domains $D_1 = \{a, b, c\}$ and $D_2 = \{d, e, f\}$, shown below.

$$\begin{tikzcd} 1 \arrow[rr, "R_{12} = \begin{bmatrix} 0\,1\,1 \\ 0\,1\,0 \\ 1\,1\,0 \end{bmatrix}"] & & 2 \end{tikzcd}$$

The constraint R_{12} does not allow, for example, the pair (a, d) but does allow the pairs (a, e), (a, f). It can be seen that the constraint has the row convexity property.

The following lemma on the intersection of (0,1)-row vectors that are row convex, is instrumental for proving the properties of interest. The lemma is a discrete version of the well-known result that for a set of mutually intersecting intervals there is a point common to all.

Lemma 8.5.4 *Let F be a finite collection of (0,1)-row vectors that are row convex and of equal length such that every pair of row vectors in F have a non-zero entry in common; that is, their intersection is not the vector with all zeroes. Then all of the row vectors in F have a non-zero entry in common.*

Proof. Let $v_1, \ldots, v_n$ be row vectors of length k. Let $\text{first}(v_i)$ and $\text{last}(v_i)$ be the position of the first and last one in v_i, respectively. For example, if $v_1 = [\ 0\ 1\ 1\ 1\ 0\ ]$, $\text{first}(v_1) = 2$ and $\text{last}(v_1) = 4$. We define $\text{first}(v_i) = k+1$ and $\text{last}(v_i) = 0$, for the zero vector of length k. We want to show that if the vectors are non-zero and their pairwise intersections are all non-zero, then the intersection of all of the vectors together is a non-zero vector. The condition that a vector, v_i, is non-zero can be expressed as, $\text{first}(v_i) \leq \text{last}(v_i)$. The condition that every pair of vectors have a non-zero intersection can be expressed as, $\forall i \forall j (\text{first}(v_i) \leq \text{last}(v_j))$. It follows that, $\max(\text{first}(v_1), \ldots, \text{first}(v_n)) \leq \min(\text{last}(v_1), \ldots, \text{last}(v_n))$, which is equivalent to saying that the intersection of all of the vectors together is a non-zero vector. $\Box$

Theorem 8.5.5 *Let $\mathcal{R}$ be a path consistent binary constraint network. If there exists an ordering of the domains $D_1, \ldots, D_n$ of $\mathcal{R}$ such that the relations are row convex, the network is globally consistent.*

Proof. The theorem is proved by showing that if the network is path consistent and all of the (0,1)-matrices are row convex, then the network is k-consistent for all $k \leq n$. Suppose that variables $x_1, \ldots, x_{k-1}$ can be consistently instantiated. That is, let $a_1, \ldots, a_{k-1}$ be an instantiation satisfying all the relevant constraints. To show that the network is k-consistent, we must show that there exists at least one instantiation, a_k, of variable x_k that is not violated by any R_{ik} $i = 1, \ldots, k-1$. The $a_1, \ldots, a_{k-1}$ restrict the allowed instantiations of x_k. For each i, the non-zero entries in row a_i of the (0,1)-matrix R_{ik} are the allowed instantiations of x_k. The key is that all of these row vectors are row convex, i.e., the ones are consecutive. Hence, by Lemma 8.5.4 it is sufficient to show that any two row vectors have a non-zero entry in common to show that they all have a non-zero entry in common. But arc consistency guarantees that each row vector contains at least one non-zero entry and path consistency guarantees that each pair of row vectors has a non-zero entry in common. Hence, all of the constraints have a non-zero entry in common and there exists at least one consistent instantiation of x_k that satisfies all the constraints. Hence, we have shown that, for any consistent instantiation of $k-1$ variables, there exists an instantiation of any kth variable. Hence, the network is k-consistent. $\Box$

Example 8.5.6 We use the familiar scene labeling example to illustrate the application of Theorem 8.5.5. Figure 8.11 shows the variables in the constraint network and the constraints; Figure 8.10 shows the domains of the variables and the ordering imposed. The constraints between variables are simply that, if two variables share an edge, then the edge must be labeled the same at both ends. Not all of the constraints are row convex. However, once the path consistency algorithm is applied, the relations become row convex. Therefore, in this example, no reordering of the domains is needed in order to satisfy the theorem. *The four possible

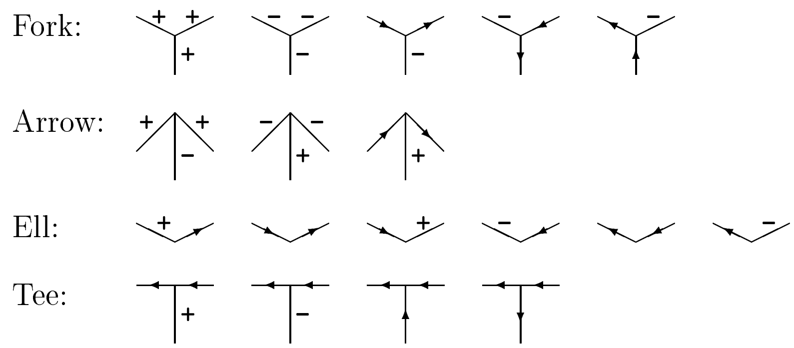
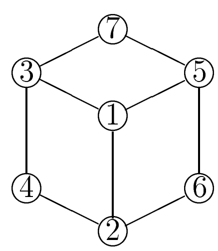


Figure 8.10: Huffman-Clowes junction labelings

$$R_{21} = \begin{bmatrix} 0\,1\,1\,0\,0 \\ 1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0 \end{bmatrix} \; R_{31} = \begin{bmatrix} 0\,1\,0\,1\,0 \\ 1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0 \end{bmatrix} \; R_{51} = \begin{bmatrix} 0\,1\,0\,0\,1 \\ 1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0 \end{bmatrix}$$



$$R_{24} = R_{37} = R_{56} = \begin{bmatrix} 0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0\,1 \\ 0\,0\,0\,1\,1\,0 \end{bmatrix}$$

$$R_{26} = R_{34} = R_{57} = \begin{bmatrix} 1\,0\,0\,0\,0\,0 \\ 0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,1\,1 \end{bmatrix}$$

Figure 8.11: Scene labeling constraint network

The scene-labeling problem has been shown to be NP-complete in the general case [91]. We are attempting to prove the conjecture that constraint networks arising from orthohedral scenes are row convex once path consistency is applied.

As an immediate corollary of Theorem 8.5.5, if we know that the result of applying path consistency will be that all of the relations will be row convex, we can guarantee *a priori* that path consistency will find the corresponding minimal network and that the minimal network will be globally consistent. Thus, if the relations in our constraint network are row convex and remain row convex under extended composition, the result applies.

Corollary 8.5.7 *Let $\mathcal{L}$ be a set of (0,1)-matrices closed under composition, intersection, and transposition such that each element of $\mathcal{L}$ is row convex. Let $\mathcal{R}$ be a binary constraint network with all relations taken from $\mathcal{L}$. The path consistency algorithm will correctly determine the minimal network of $\mathcal{R}$. Furthermore, the minimal network will be globally* *consistent.*

Proof. This can be proved by a simple rewriting of the proof for Theorem 8.5.5. □

Example 8.5.8 Let the domains of the variables be finite subsets of the integers and let a binary constraint between two variables be a conjunction of linear equalities and inequalities of the form $ax_i - bx_j = c$, $ax_i - bx_j < c$, or $ax_i - bx_j \leq c$, where a, b, and c are integer constants. For example, the conjunction

$$(3x_i + 2x_j \leq 3) \wedge (-4x_i + 5x_j < 1)$$

is an allowed constraint between variables x_i and x_j. A network with constraints of this form can be formulated as an integer linear program where each constraint is on two variables and the domains of the variables are restricted to be finite subsets of the integers. However, it can be shown that each element in the closure under composition, intersection, and transposition of the resulting set of (0,1)-matrices is row convex, provided that when an element is removed from a domain by arc consistency, the associated (0,1)-matrices are "condensed." This is best illustrated through an example. Let $D_i = D_j = \{-1, 0\}$ and $D_k = \{-1, 0, 1\}$ and let R_{ij} be the matrix constructed from the constraint $x_i + x_j < 0$ and R_{jk} be constructed from $-2x_j + x_k = 1$.

$$R_{ij} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad R_{jk} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R_{ik} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

The matrix R_{ik}, the result of composing R_{ij} and R_{jk}, is not row convex. However, there is no solution with x_k assigned 0, so D_k becomes $\{-1, 1\}$ and row convexity can be restored by removing the middle column from matrices R_{ik} and R_{jk}. Hence, we can guarantee that the result of path consistency will be the minimal network and the network will be globally consistent. □

8.5.1 Identifying Row-Convex Relations

As noted in the scene-labeling example, when constructing a constraint network and the (0,1)-matrices that represent the constraints, we must impose an ordering on the domains of the variables. Sometimes a natural ordering exists, as when the domain is a finite subset of the integers, but often the ordering imposed is arbitrary and with no inherent meaning. An unlucky ordering may hide the fact that the constraint network really is row convex or, more properly, can be made row convex. How can we distinguish this case from the case where no ordering of the domains will result in row convexity? The following theorem shows that we can test for this property efficiently.

Theorem 8.5.9 (Booth and Lueker [17]) *An $m \times n$ (0,1)-matrix specified by its f nonzero entries can be tested for whether a permutation of the columns exists such that the matrix is row convex in $O(m + n + f)$ steps.*

Example 8.5.10 Maruyama [117] shows that natural language parsing can be formulated as a problem on constraint networks. In this framework, intermediate parsing results are represented as a constraint network and every solution to the network corresponds to an individual parse tree. We use an example network from [117] to illustrate the application of Theorems 8.5.5 and 8.5.9. Consider the following sentence.

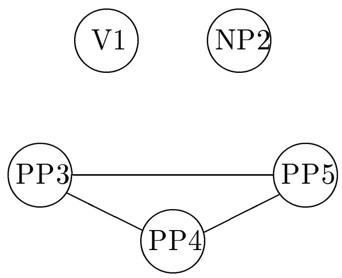
Put	the block	on the floor	on the table	in the room.
V1	NP2	PP3	PP4	PP5

The sentence is structurally ambiguous (there are fourteen different parses) as there are many ways to attach the prepositional phrases. Figure 8.12 shows the original ordering of the domains; Figure 8.13 shows the variables in the constraint network and the constraints. For example, the constraint between variable PP3 and variable PP4 is given by the (0,1)-matrix at row PP3 column PP4 of the table in Figure 8.13 (the symbol I in the figure denotes the identity matrix—the (0,1)-matrix consisting of ones along the diagonal and zeroes everywhere else). While the network is path consistent, it can be seen that the constraints are not all row convex given the original domain ordering used in [117]. However, using the new domain ordering shown in Figure 8.12, the constraints are now row convex. □

Variable	Domain	
	Original ordering	New ordering
V1	{Rnil}	{Rnil}
NP2	{O1}	{O1}
PP3	{L1, P2}	{L1, P2}
PP4	{L1, P2, P3}	{P2, P3, L1}
PP5	{L1, P2, P3, P4}	{P3, P4, L1, P2}

Figure 8.12: Variables and domains for parsing example

Let $\mathcal{R}$ be a path consistent binary constraint network. It remains to show how Theorem 8.5.9 can be used to determine whether an ordering of the domains of the variables exists such that all of the (0,1)-matrices R_{ij}, $1 \leq i, j \leq n$, are row convex. The procedure is simple: for each variable, x_j, we take the matrix defined by stacking up R_{1j} on top of R_{2j} on top of $\cdots R_{nj}$ and test whether the matrix can be made row convex. For example, with reference to Figure 8.13, for variable PP4 we would test whether the columns of the matrix consisting of the 3 columns and 11 rows under the column heading PP4 can be



	V1	NP2	PP3	PP4	PP5
V1	I	1	1 1	1 1 1	1 1 1 1
NP2	1	I	1 1	1 1 1	1 1 1 1
PP3	1	1	I	1 0 1	1 0 1 1
	1	1		1 1 1	1 1 1 1
PP4	1	1	1 1	I	1 0 0 1
	1	1	0 1		1 1 0 1
	1	1	1 1		1 1 1 1
PP5	1	1	1 1	1 1 1	I
	1	1	0 1	0 1 1	
	1	1	1 1	0 0 1	
	1	1	1 1	1 1 1	

Figure 8.13: Left: Parsing constraint network; Right: Table representation of constraint network

permuted to satisfy the row convexity property. In this example such a permutation exists and corresponds to the new ordering of the domain of variable PP4 shown in Figure 8.12.

8.5.2 Non-Binary Row-Convex Constraints

Informally, an r-ary relation is row convex if, in the multidimensional 0/1 matrix representing the constraint, each vector that is parallel to one of the axes has the consecutive 1's property.

Definition 8.5.11 *An r-ary relation R_S $S = \{x_1, \ldots, x_r\}$ is* row convex *if for any subset Z of $r-2$ variables $Z \subseteq S$ and for every instantiation, $\vec{a}$, of Z, the binary relation $\pi_{(S-Z)}(\sigma_{\vec{a}}(R))$ is row convex.*

For binary constraint networks, we identified an efficient procedure for determining whether a domain ordering exists such that the relations are all row convex (see Section 8.5.1). It is an open question whether such an efficient procedure exists for r-ary constraint networks. However, it is easy to see that some important classes of constraints such as bi-valued relations, implicational constraints, and linear inqequalities are all row convex.

We now show that general relational path-consistency ensures global consistency when the relations are row convex.

Theorem 8.5.12 *Let $\mathcal{R} = (X, D, C)$ be a network of constraints that is relationally path-consistent. If there exists an ordering of the domains $D_1, \ldots, D_n$ of $\mathcal{R}$ such that the relations are row convex, the network is globally consistent.*

Proof. Assume that the network is relationally path-consistent. Let $\vec{a}_{i-1} = (a_1, a_2, \ldots, a_{i-1})$ be a consistent instantiation of $i-1$ variables. We show that for any x_i, there exist a value a_i of x_i such that the extended tuple $(a_1, a_2, \ldots, a_{i-1}, a_i)$ is consistent. This means that any given applicable relation $R_Y \in \mathcal{R}$ that is defined over $\{x_1, \ldots, x_i\}$ should be satisfied. Since all of the constraints are relationally arc-consistent, each must be consistent with $\vec{a}_{i-1}$. Since they are also row convex, all of the values of x_i that are allowed by each R_Y and that are restricted to the values in X' can be listed as a row convex vector. Relational path-consistency implies that any two rows corresponding to any two constraints, must have a common value in their intersection. Since they are all row convex they all have a common value in their intersection and the claim is proved. □

Therefore,

Theorem 8.5.13 *For any network $\mathcal{R}$, whose closure under extended 2-composition is row convex, RC_2 will generate a globally-consistent network of $\mathcal{R}$.*

Example 8.5.14 Consider a set of r-ary linear inequalities, where the domains of the variables are finite subsets of integers and the r-ary constraints over a subset of variables $x_1, \ldots, x_r$ are of the form $a_1x_1 + \cdots + a_rx_r \leq c$, where the a_i's are rational constants. This is a general integer linear program. The r-ary inequalities define corresponding r-ary relations that are row convex. Therefore, whenever the extended composition of any two of these r-ary relations can be guaranteed to be row-convex their consistency can be determined by RC_2. □

Example 8.5.15 Consider now the special case of the following linear constraints over the *non-negative* integers:

$$(1)\ f + x + y + z \leq 1, \qquad (2)\ f - z \leq -1$$

The constraints are not relationally arc-consistent. For instance, $f = 1, z = 2$ satisfies (2), and thus is in $\rho(f, z)$, but it cannot be extended to (1). In order to make constraint (1) relationally arc-consistent we have to add all of its projections having the form: for any subset $T \subset \{f, x, y, z\}$ $\sum_{x_i \in T} x_i \leq 1$. To make (2) relationally arc-consistent we have to add only $z \geq 1$. Once these sets of linear inequalities are added to (1) and (2) we have relational arc-consistency. Still we do not have relational path-consistency. For instance, the instantiation $f = 0, x = 1, y = 0$ satisfies all of the constraints, but it cannot be consistently extended to a value satisfying (1) and (2). If we add the constraint (3) $2f + x + y \leq 0$, constraints (1) and (2) will become relationally path-consistent relative to z since constraint (3) will disallow the partial assignments $f = 0, x = 1, y = 0$. Constraints (1) and (2) are also relationally path-consistent relative to f since any consistent instantiations of x, y, and z will have to satisfy the two constraints $x + y + z \leq 1$

and $-z \leq -1$ that were added to make the network relationally arc-consistent. Once these constraints are obeyed there is an extension to $f = 0$ that satisfies (1) and (2) simultaneously.

Thus if all the constraints created are row-convex they express a globally consistent network. □

Clearly,

Theorem 8.5.16 (completeness) *For any network $\mathcal{R} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$, whose directional closure relative to extended 2-composition is not empty and row convex, algorithm DRC_2 computes an equivalent network of $\mathcal{R}$ that is backtrack-free along the ordering d.*

In the next section we will see that DRC_2 specialize to a well known algorithm for processing linear inequalities.

8.6 Linear inequalities

In this section, we note the applicability of consistency enforcing to the special case of linear inequalities over infinite domains like the Rationals, as well as over finite and infinite subsets of the Integers. In database theory, a k-ary relation r is a finite set of tuples, and a database is a finite set of relations. However, the relational calculus and relational algebra can be developed without the assumptions of finiteness for relations. We will use the term *unrestricted relation* for finite or infinite sets of points in a k-dimensional space.

Let us consider the case where a constraint between r or fewer variables is a linear inequality of the form $\sum_{i=1}^{r} a_i x_i \leq c$, where a_i, and c are Rational constants.[1] For example, the conjunction $(3x_i + 2x_j \leq 3) \wedge (-4x_i + 5x_j \leq 1)$ includes two constraints between variables x_i and x_j. We show that the standard operation of variable elimination for this class of expressions, is equivalent to extended 2-composition. Let us denote by $sol(\alpha)$ the unrestricted relation of tuples from the domain satisfying a set of linear inequalities, α. We define the elimination operation as follows:

Definition 8.6.1 ((linear elimination)) *Let $\alpha = \sum_{i=1}^{(r-1)} a_i x_i + a_r x_r \leq c$, and $\beta = \sum_{i=1}^{(r-1)} b_i x_i + b_r x_r \leq d$. Then $elim_r(\alpha, \beta)$ is applicable only if a_r and b_r have opposite signs, in which case $elim_r(\alpha, \beta) = \sum_{i=1}^{r-1} (-a_i \frac{b_r}{a_r} + b_i) x_i \leq -\frac{b_r}{a_r} c + d$. If a_r and b_r have the same sign, the elimination implicitly generates the universal constraint.*

[1] If we insist on a unique constraint for a given subset of variables in a scheme, we would have to define a constraint as a conjunction of linear inequalities, all defined on the same set of variables.

In principle, if the domains are finite, we can convert each linear inequality to a relational representation and then enforce relational consistency. Alternatively, we can apply linear elimination directly to the mathematical expressions.

The next lemma says that extended 2-composition is identical to linear elimination over the Rationals, while it is not quite so over the Integers.

Lemma 8.6.2 *$sol(elim_r(\alpha,\beta)) \supseteq EC_r(sol(\alpha), sol(\beta))$ when the domains are the Integers. However, over the Rationals $sol(elim_r(\alpha,\beta)) = EC_r(sol(\alpha), sol(\beta))$.*

Proof It is easy to see that if a_r and b_r have the same sign (both are positive or both are negative), then for any assignment to $x_1,\ldots,x_{i-1}$, there is always a value for x_r that extends $x_1,\ldots,x_{i-1}$ that satisfies both α and β. Therefore, the extended composition produces the universal relation. Assume now that a_r and b_r have opposite signs. Multiplying α by $-\frac{b_r}{a_r}$ and summing the resulting inequality with β yields the inequality

$$\sum_{i=1}^{r-1}(-a_i\frac{b_r}{a_r}+b_i)x_i \leq -\frac{b_r}{a_r}c+d.$$

In other words, any tuple satisfying this inequality can be extended to a *rational value* of x_r in a way that satisfies both α and β. It is unclear, though, that there exists an integer extension to x_r, which is the reason for partial containment for the integers. $\Box$

Linear elimination that is applied when one of the constraints is a domain constraint of the type $x \leq a$ can yield both domain propagation and relational arc-consistency, and both rules are popular in processing systems of linear inequalities.

8.6.1 Fourier Bucket elimination

Incorporating linear elimination into DRC_2, when the constraints are specified as linear inequalities yields in algorithm *Directional Linear Elimination* (abbreviated DLE) which is the well known Fourier elimination algorithm.

Like in the case of propositional theories, DRC_2 generates a directionally globally consistent problem. The algorithm decides the solvability of any set of linear inequalities over the Rationals and generates a problem representation which is backtrack-free. The algorithm, is summarized in Figure 8.14.

Theorem 8.6.3 *Given a set of linear inequalities φ, algorithm DLE (Fourier elimination) decides the consistency of such a set of linear inequalities over the Rationals and the Reals, and it generates an equivalent backtrack-free representation.* $\Box$

DIRECTIONAL-LINEAR-ELIMINATION (φ, d)
Input: A *set of linear inequalities* φ, an ordering $d = x_1, \ldots, x_n$.
Output:A decision of whether φ is satisfiable. If it is, a backtrack-free theory $E_d(\varphi)$.
1. **Initialize:** Partition inequalities into ordered buckets.
2. **for** $i \leftarrow n$ **downto** 1 **do**
3. **if** x_i has one value in its domain **then**
. substitute the value into each inequality in the bucket and put the resulting inequality in the right bucket.
4. **else,for each** pair $\{\alpha, \beta\} \subseteq bucket_i$, compute $\gamma = elim_i(\alpha, \beta)$
if γ has no solutions, return $E_d(\varphi) = \{\}$, "inconsistency"
else add γ to the appropriate lower bucket.
5. **return** $E_d(\varphi) \leftarrow \bigcup_i bucket_i$

Figure 8.14: Fourier Elimination; DLE

$bucket_4$: $5x_4 + 3x_2 - x_1 \leq 5,\ x_4 + x_1 \leq 2,\ \ - x_4 \leq 0,$
$bucket_3$: $x_3 \leq 5,\ x_1 + x_2 - x_3 \leq -10$
$bucket_2$: $x_1 + 2x_2 \leq 0.$
$bucket_1$:

Figure 8.15: initial buckets

Proof: The proof can be given directly (see exercise). It also follows from the fact that linear inequalities can be shown to be *row-convex* a property that, as we show in a later chapter, necessitates only relational 2-consistency for global consistency.

The complexity of Fourier elimination is not bounded exponentially by the induced width, however. The intuitive reason is that the number of linear inequalities that can be specified over a scope of size i cannot be bounded exponentially by i.

Example 8.6.4 Consider the set of inequalities over the Reals:

$$\varphi(x_1, x_2, x_3, x_4) = \{(1)\ 5x_4 + 3x_2 - x_1 \leq 5,\ (2)\ x_4 + x_1 \leq 2,\ (3)\ \ - x_4 \leq 0,$$
$$(4)\ x_3 \leq 5,\ (5)\ x_1 + x_2 - x_3 \leq -10,\ (6)\ x_1 + 2x_2 \leq 0\}.$$

The initial partitioning into buckets is shown in Figure 8.15. Processing $bucket_4$, which

$bucket_4$: $5x_4 + 3x_2 - x_1 \leq 5,\ x_4 + x_1 \leq 2,\ \ -x_4 \leq 0,$
$bucket_3$: $x_3 \leq 5,\ x_1 + x_2 - x_3 \leq -10$
$bucket_2$: $x_1 + 2x_2 \leq 0 \mid\mid 3x_2 - x_1 \leq 5, x_1 + x_2 \leq -5$
$bucket_1$: $\mid\mid x_1 \leq 2.$

Figure 8.16: final buckets

involves applying elimination relative to x_4 over inequalities {(1),(3)} and {(2),(3)}, respectively, results in $3x_2 - x_1 \leq 5$, placed into $bucket_2$, and $x_1 \leq 2$, placed into $bucket_1$. Processing the two inequalities in $bucket_3$ next, $x_3 \leq 5$, and $x_1 + x_2 - x_3 \leq -10$, eliminates x_3, yielding $x_1+x_2 \leq -5$, placed into $bucket_2$. When processing $bucket_2$, containing $x_1 + 2x_2 \leq 0$, $3x_2 - x_1 \leq 5$, and $x_1 + x_2 \leq -5$, no new inequalities are added. The final set of buckets is displayed in Figure 8.16. □

We can now generate a backtrack-free solution in the following way. Select a value for x_1 from its domains satisfying the unary inequalities in $bucket_1$. From there on, after selecting assignments for $x_1, \ldots, x_{i-1}$, select a value for x_i that satisfies all the inequalities in $bucket_i$. This is an easy task, since all the constraints are unary once the values of $x_1, \ldots, x_{i-1}$ are determined.

8.7 Summary

This chapter extends consistency definitions and algorithms from variable-based to relation-based which allow expressing new types of bounded inference, not captures by variable-based i-consistency. The concepts of relational arc- path- and i-consistency are defined. These definitions capture inference operators for languages such as propositional theories and algebraic linear inequalities. Extension of relational consistency to directional relational consistency follows naturally and is described within the bucket-elimination formulation. Specializing DRC_1 and DRC_2 to cnf formulas and to linear inequalities yields well known variable elimination algorithms such as the Davis-Putnam algorithm (which we call directional-resolution) and Fourier elimination. We also addressed explicitly well known tractable classed such as 2-SAT and Horn theories and placed those within the general relational framework.

8.8 Chapter notes

Relational-consistency were defined by van Beek and Dechter in a sequence of three papers [170, 167, 169, 46]. They proved bounded domain, constraint tightness and row-convexity if accompanied with corresponding bounded levels of local consistency guarantee global consistency. Relational-consistency allowed expressing these relationships for non-binary networks in a natural way. Some of these results will be elaborated in later chapters. The results on domain-size and consistency was presented by Dechter [39]. The other two relationships can be stated as follows: If the constraints are m-tight and relational m-consistent then they are globally consistent []. Additionally if the constraints are row-convex and relational 2-consistent they are globally consistent [166].

The observation that relational 2-consistency and adaptive-consistency i general implies algorithm directional-resolution for cnfs was presented in [45, 140]. It was observed that the resulting algorithm is the well known David-Putnam algorithm. Indeed this algorithm was immediately replaced with a backtracking style algorithm due to the heave space complexity. The analysis of the complexity of this algorithm as a function of w* was then derived.

Similarly, applying DRC_2 to algebraic equations yielded a variable elimination algorithm that was later observed to be the well known Fourier elimination [104].

8.9 Exercises

1. Consider the constraint network over the set of variables $\{x_1, x_2, x_3, x_4, x_5\}$, where each variable has the domains $D_i = \{$a,b,c$\}$, $1 \leq i \leq 5$, and the relations are given by

   $$\begin{aligned} R_{2,3,4,5} &= \{ \text{(a,a,a,a), (b,a,a,a), (a,b,a,a), (a,a,b,a), (a,a,a,b)} \}, \\ R_{1,2,5} &= \{ \text{(b,a,b), (c,b,c), (b,a,c)} \} \end{aligned}$$

   Are the constraint relational arc and path consistent? If not makes the network relational arc and path consistent.

2. Prove that resolution is equivalent to extended 2-composition

3. Prove that algorithm DLE can decide the consistency of a set of linear inequalities over the Rationals and the Reals.

4. Analyze the complexity of DLE over finite domains and over Rationals.

5. Prove that: Algorithm DLE is polynomial over the class of unary and binary inequalities of the form $x - y \leq a$, $X \leq b$. Prove also that the algorithm directionally globally solves such inequalities, over the Integers (if a and b are integers), the Rationals and the Reals.

6. Linear inequalities over the integers:

   (a) Over Integer domains algorithm DLE is no longer guaranteed to decide consistency since linear elimination is not identical to extended 2-composition. If the empty relation is generated by DLE, can you conclude that the problem is inconsistent? explain your answer.

   (b) Would a backtrack-algorithm be backtrack-free if applied following DLE? will it have a smaller search space?

7. We next present (Figure 8.17) a directional algorithm that allow bounding the complexity of inference both by the number of variables and by the number of constraints. Prove that the complexity of $DRC_{(i,m)}$ is $O(n^i(k \cdot n)^{(im)})$ where k bounds the domain sizes and n is the number of variables.

8. Consider the following cnf theory

$$\varphi = \{(a, \neg b, d), (\neg d, f, c), (\neg c)\}$$

   1. Define RC_1 and RC_2 for CNF theories using the resolution operator. 2. Apply RC_1 to this theory. Apply RC_2 to this theory.

9. Consider the following set of cnf formulas: xxx Show the execution of algorithm directional-resolution on this theory along ordering d_1 and d_2. Bound the complexity of the algorithm along each ordering.

10. Consider the following set of linear inequalities: xxx. Show the execution of algorithm DLE on this theory along ordering d_1 and d_2. Bound the complexity of the algorithm along each ordering.

11. Prove that a theory that includes all its prime implicates is globally consistent.

DIRECTIONAL-RELATIONAL-CONSISTENCY($\mathcal{R}, i, m, o$) ($DRC_{(i,m)}$)

1. Initialize: generate an ordered partition of the constraints, $bucket_1, \ldots, bucket_n$, where $bucket_i$ contains all constraints whose highest variable is x_i.
2. **for** $p \leftarrow n$ **downto** 1 **do**
3.   simplification step:
    for every $S_i, S_j \in bucket_p$, such that $S_i \supseteq S_j$ **do**
        $R_{S_i} \leftarrow \pi_{S_i}(R_{S_i} \bowtie R_{S_j})$
4.   instantiation step:
    if $bucket_p$ contains the constraint $x_p = u$ **then**
        for every $S_i \in bucket_p$ **do**
            $A \leftarrow S_i - \{x_p\}$
            $R_A \leftarrow \pi_A(\sigma_{x_p=u} R_{S_i})$
                if R_A is not the empty relation **then**
                    add R_A to its appropriate bucket
                else exit and return the empty network
5.   **else** (the general case)
        $j \leftarrow \min\{\text{cardinality of bucket}_p,\ m\}$
6.   **for** every j relations $R_{S_1}, \ldots, R_{S_j}$ in $bucket_p$ **do**
            $F_j \leftarrow \bowtie_{t=1}^{j} R_{S_t}$
7.   **for** every subset A of size $iA \subseteq \bigcup_{t=1}^{j} S_t - \{x_p\}$ **do**
8. $R_A \leftarrow \pi_A F_j$
9.   **if** R_A is not the empty relation **then**
10.   add R_A to its appropriate bucket
11.   **else** exit and return the empty network
12. **return** $E_{(i,m)}(R) = \bigcup_{j=1}^{n} bucket_j$

Figure 8.17: Algorithm $DRC_{i,m}$