# Answer Set Optimization for and/or Composition of CP-Nets: A Security Scenario*

Stefano Bistarelli[1,2], Pamela Peretti[1], and Irina Trubitsyna[3]

[1] Dipartimento di Scienze, Università "G. d'Annunzio", Pescara, Italy
{bista,peretti}@sci.unich.it

[2] Istituto di Informatica e Telematica, CNR, Pisa, Italy
Stefano.Bistarelli@iit.cnr.it

[3] DEIS Università della Calabria, Rende, Italy
irina@deis.unical.it

**Abstract.** Defence trees are used to represent attack and defence strategies in security scenarios; the aim in such scenarios is to select the *best* set of countermeasures have to be applied to stop all the vulnerabilities. To represent the preference among the possible countermeasures of a given attack, defence trees are enriched with CP-networks (CP-net for short). However, for complex trees, composing CP-nets could be not always effective. In this paper we overcome these limitations by transforming each CP-net in an *Answer Set Optimization* (ASO) program. The ASO program, representing the overall scenario, is a special composition of the programs associated to each branch of the defence tree. The best set of countermeasure able to mitigate all the vulnerabilities is then obtained by computing the optimal answer set of the corresponding ASO program.

## 1 Introduction

Security has become today a fundamental part of the enterprise investment. In fact, more and more cases are reported showing the importance of assuring an adequate level of protection to the enterprise's assets. In order to focus the real and concrete threat, a risk management process is needed to identify, describe and analyze the possible vulnerabilities that have to be eliminated or reduced. The final goal of the process is to make security managers aware of the possible risks, and to guide them toward the adoption of a set of countermeasures which can bring the overall risk under an acceptable level.

*Defence trees*, DT [3], have been introduced as a methodology for the analysis of attack/defence security scenarios. A DT is an *and-or* tree, where leaves node represent the vulnerabilities and the set of countermeasures available for their mitigation, `and` nodes represent attacks composed by a set of actions that have to be performed as a whole, and `or` nodes represent attacks that can succeed also by completing only one of their child action. Notice that to defeat `and` attacks it is enough to patch one of the vulnerabilities (by selecting a single

* This paper is partially supported by the MIUR PRIN 2005-015491.

C. Bessiere (Ed.): CP 2007, LNCS 4741, pp. 773–781, 2007.

countermeasure), whilst to stop `or` attacks, one countermeasure for each of the actions composing the attack has to be selected.

The overall goal is to use the defence tree representation for the selection of the best set of countermeasures (w.r.t. specific preference criteria such as cost, efficiency, etc.), that can stop all the attacks to the system. To guide the selection, *CP-nets* [5] have been proposed to model preferences over attacks and countermeasures [4]. However, CP-nets are a graphical formalism that is good and elegant for representing small scenarios, but not so effective for big and complex scenarios. In particular, the methodology proposed in [4] aims at composing together the CP-nets associated to each branch of the tree. Each CP-net, in turn, represents the preferences among the countermeasure able to stop a specific attack.

In this paper the preference among countermeasures and the dependency between attacks and countermeasures, represented as a CP-net, are translated in *answer set optimization* (ASO) [8] programs. The `and` and `or` composition of the branch is then obtained by a syntactic composition of the ASO programs, whose semantics completely respects the intended meaning given in [4]. The semantics of the obtained ASO program provides a set of ordered answer sets representing the ordered sets of countermeasure to be adopted. To deal with ordered attacks (from the more to the less dangerous), the model is extended by introducing a corresponding rank among the preference rules of an ASO program. In this case, a preference cycle among countermeasure could be generated in the resulting CP-net. The use of ranked ASO program avoids this problem; in fact, the introduction of meta-preferences gives precedence to the adoption of countermeasures covering the more dangerous of the attacks (and removing in this way the possibility to obtain a cycle).

## 2 Defence Tree

*Defence trees* [3] are an extension of attack trees [12] and are used to represent attack strategies that can be used a mitigation factor.

The root of the tree is associated with an asset of the IT system under consideration and represents the attacker's goal. Leaf nodes in the attack tree represent simple subgoals which lead the attacker to damage the asset by exploiting a single vulnerability. Non-leaf nodes (including the tree root) can be of two different types: `or`-nodes and `and`-nodes. Subgoals associated with `or`-nodes are completed as soon as any of its child nodes is achieved, while `and`-nodes represent subgoals which require all of its child nodes to be completed (in the following we draw an horizontal line between the arcs connecting an `and`-node to its children to distinguish it from an `or`-node). The standard attack tree is then enriched by decorating every leaf node with a set of countermeasures. Each countermeasure associated with a leaf node represents a possible way of mitigating risk in an attack scenario where that specific vulnerability is used.

Notice that in order to mitigate the risks deriving from an `or`-attack, the system administrator has to introduce into the system *a countermeasure for*

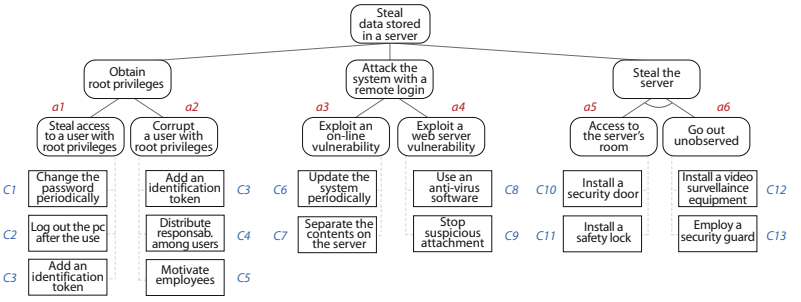



**Fig. 1.** An example of defence tree

*each possible action of the branch*. To mitigate the risks associated with an `and`-attack, instead, it *is enough to introduce a countermeasure for one of the attack actions* in the `and`-attack to stop the entire attack.

In the following example we use a defence tree to model an attack/defence scenario for an organization's IT system.

*Example 1.* An enterprise's server is used to store information about customers. Figure 1 shows the corresponding attack/defence scenario: rounded-box nodes denote the attack strategies and the different actions the attacker needs to perform, while square box denote the different countermeasures the system administrator can adopt. △

## 3 Answer Set Optimization Programs

Prioritized reasoning is an important extension of logic programming, that introduce preference on partial and complete solutions. A variety of approaches was proposed in the literature (see [10] for a survey of this topic). In this work we uses two approaches for representing and reasoning with preference statements: ASO and CR-prolog$_2$.

The ASO approach uses preference rules in order to express the preference relations among the combinations of atoms and introduces the preference order among these rules. Moreover a tool implementing the ASO semantics and its extensions, CHOPPER, has been recently proposed in [9].

An alternative way consists in the use of CR-prolog [1], a knowledge representation language based on the answer set semantic enriched with the introduction of consistency-restoring rules that allows a formalization of events or exceptions that are unlikely, unusual, or undesired. Its extension, CR-prolog$_2$ [2] admits the ordered disjunction in the head of rules and can be used to represent preferences intended as strict preferences (as in CR-prolog) or desires (LPOD [7]).

An answer set optimization program (ASO) is a pair $\langle \mathcal{P}, \Phi \rangle$, where $\mathcal{P}$ is an answer set program [11], called *Generating Program*, and $\Phi$ is called *Preference Program* and consists of a finite set of preference rules of the form $\varrho : \ C_1 > \cdots > C_k \leftarrow body$, where *body* is a conjunction of literals, i.e. atoms or negation of atoms, and $C_i$s are *boolean combinations* of literals, i.e formulas built of atoms

by means of disjunctions, conjunctions, strong ($\neg$) and default (*not*) negation with the restriction that strong negation is allowed to appear only in front of atoms and default negation only in front of literals.

A preference rule $\varrho \in \Phi$ introduces a preference order between $C_1, ..., C_k$: $C_i$ is preferred to $C_j$, for $i < j$ and $i, j \in [1..k]$. Thus $\Phi$ determines a preference ordering on the answer sets described by $\mathcal{P}$.

Let $\Phi = \{\varrho_1, ..., \varrho_n\}$ be a preference program and $S$ be an answer set, then $S$ induces a *satisfaction vector* $V_S = (v_S(\varrho_1), ..., v_S(\varrho_n))$ where: a) $v_S(\varrho_j) = I$, if $\varrho_j$ is *Irrelevant* to $S$, i.e. (i) the body of $\varrho_j$ is not satisfied in $S$ or (ii) the body of $\varrho_j$ is satisfied, but none of the $C_i$s is satisfied in $S$. b) $v_S(\varrho_j) = min\{i : S \models C_i\}$, otherwise. The satisfaction vectors are used to compare the answer sets.

Let $S_1$ and $S_2$ be two answer sets, then (i) $S_1 \geq S_2$ if $V_{S_1} \leq V_{S_2}$, i.e. if $v_{S_1}(\varrho_i) \leq v_{S_2}(\varrho_i)$ for every $i \in [1..n]$; (ii) $S_1 > S_2$ if $V_{S_1} < V_{S_2}$, i.e. if $V_{S_1} \leq V_{S_2}$ and for some $i \in [1..n]$ $v_{S_1}(\varrho_i) < v_{S_2}(\varrho_i)$.
A set of literals $S$ is an *optimal answer set* of an ASO program $\langle \mathcal{P}, \Phi \rangle$ if $S$ is an answer set of $\mathcal{P}$ and there is no answer set $S'$ of $\mathcal{P}$ such then $S' > S$.

The ASO strategy is further extended by introducing *meta-preferences* among preference rules: a ranked ASO program is a sequence $\langle \mathcal{P}, \Phi_1, ..., \Phi_n \rangle$ consisting of a generating program $\mathcal{P}$ and a sequence of pairwise disjoint preference programs $\Phi_i$. The rank of a rule $r \in \Phi_1 \cup \cdots \cup \Phi_n$, denoted $rank(r)$, is the unique integer $i$ for which $r \in \Phi_i$.

Intuitively, the rank of the preference rule determines its importance: preference rules with lower rank are preferred over preference rules with higher rank. The optimal answer sets can be obtained in the following way: firstly, all answer sets optimal w.r.t. $\Phi_1$ have to be selected; then, have to be selected the ones optimal w.r.t. $\Phi_2$; and so on. More formally, $S_1 \geq_{rank} S_2$ if for every preference rule $r'$ such that $v_{S_1}(r') \leq v_{S_2}(r')$ does not hold, there is a rule $r''$ such that $rank(r'') < rank(r')$ and $v_{S_1}(r'') < v_{S_2}(r'')$.

## 4 CP-Defence Trees

CP-networks [5] are a graphical formalism for representing and reasoning with preference statements which proposes a *ceteris paribus* (all else being equal) interpretation. The combination of defence trees and CP-nets has been recently proposed [4] as a methodology to help the system administrator to analyze a security scenario and to give him a model to represent preferences among countermeasure.

The resulting structure, called *CP-defence tree*, integrating the CP-net described in Figure 2 is presented in Figure 2(c). The CP-net graph $G$ in Figure 2(a) highlights that the preference over countermeasures is conditioned by the corresponding attack. $CPT(A)$ describes the preference of protecting from each attack $a_i \in A$: the system administrator prefers to protect the action $a_2$ to the action $a_1$, $a_1$ to the action $a_6$ and so on. $CPT(C)$ collects the preferences among the countermeasures, that protect each action. For instance, for the action $a_3$ the countermeasure $c_6$ is preferable to $c_7$.

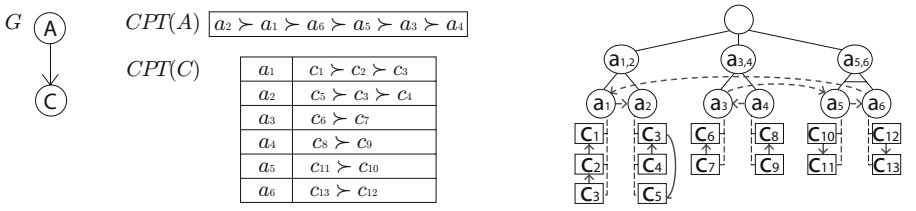



**Fig. 2.** A CP-defence tree

Considering Figure 2(c) the preference order over actions, described in $CPT(A)$, is represented with dotted arrows; while conditional preferences over countermeasures, described in $CPT(C)$, are represented by using solid arrows. The arrows are directed from the less preferred to the more preferred outcome.

Thus, given an IT system represented as the root of the tree, the corresponding CP-defence tree gives a graphical representation of the attack strategies that an attacker can pursue to damage the system, the different actions composing each attack and the countermeasures to be activated to stop them. Moreover, a preference order among attacks and countermeasures is highlighted.

### 4.1 *and/or* Composition of Attacks

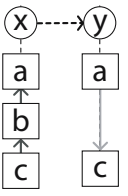
In order to find the preferred defence strategy, the approach, consisting in the composition of preferences, was proposed in [4]. More in details, two different methods establishing the preference order among the countermeasures able to stop an `and`-attack and an `or`-attack were presented.

**`and`-composition.** To protect the system from an attack obtained by an `and`-composition of actions, the system administrator can stop any one of them. When an order among attacks is given the resulting strategy consist in the selection of the best countermeasure for the most dangerous of the actions.

Sometime not only the best countermeasure have to be activated but some of them (for instance because the countermeasure is only able to cover part of the attack). In this case the system administrator need to consider not only the best countermeasure, but the overall resulting countermeasure order.

More formally, given the `and`-attack, composed by two actions $u$ and $v$, where $u \succ v$, and given the sets of countermeasures $D_u$ and $D_v$, protecting from $u$ and $v$ respectively, and two partial order $(D_u, \succ_u)$ and $(D_v, \succ_v)$, the ***`and`****-composition* of preferences is a new partial order $(D_u \cup D_v, \succ_{v \wedge v})$, where a countermeasure $c$ is preferred to a countermeasure $c'$ if (i) it is preferred in at least one of the partial orders $(D_u, \succ_u)$, $(D_v, \succ_v)$, i.e. $c \succ_u c'$ or $c \succ_v c'$, otherwise (ii) $c$ is the worst countermeasure in $(D_u, \succ_u)$, while $c'$ is the best countermeasure in $(D_v, \succ_v)$, i.e $\forall x \in D_u$, $x \not\succ_u c$ and $\forall y \in D_v$, $c' \not\succ_v y$.

Thus, the `and`-composition, corresponding to the `and`-attack, preserves the partial orderings among the countermeasures, corresponding to each attack ac-




$\mathcal{P}_{and}$ $\mathtt{r_1}$ : root ←
$\mathtt{r_2}$ : x ∨ y ← root
$\mathtt{r_{x_2}}$ : a ∨ b ∨ c ← x
$\mathtt{r_{y_2}}$ : a ∨ c ← y
$\varrho_{\mathtt{x}}$ : a > b > c ← x
$\varrho_{\mathtt{y}}$ : c > a ← y

$\mathcal{P}_{or}$ $\mathtt{r_1}$ : root′ ←
$\mathtt{r_2}$ : x ← root′
$\mathtt{r_3}$ : y ← root′
$\mathtt{r_{x_2}}$ : a ∨ b ∨ c ← x
$\mathtt{r_{y_2}}$ : a ∨ c ← y
$\varrho_{\mathtt{x}}$ : a > b > c ← x
$\varrho_{\mathtt{y}}$ : c > a ← y

**Fig. 3.** An example of and/or attacks (with cycle) and the corresponding ASO programs

tion and introduces the *bridge preference rule*, connecting the corresponding orderings. As an example consider the defence tree in the left side of Figure 3, and consider an `and`-attack *root*, composed by two actions $x$ and $y$, where $y \succ x$. The order obtained by `and`-composition is $c \succ a \succ b \succ c$. We can notice that in this case a cycle is obtained. Since the countermeasure of the worst attacks have to be considered as preferred, the cycle is broken by removing one of the preference among the countermeasure of the less dangerous attack $x$. More precisely, the preference relations, described in $(D_y, \succ_y)$ has to be considered as more important, and the relation $b \succ_x c$, generating (transitively) the conflict, has to be omitted.

As shown [6] not always is possible determine the most preferred outcome from a CP-net with an inconsistent preference order (i.e. cycle). In order to solve this problem we use the preference order over attacks in order to delete some edges from the induced preference graph and break the cycle. If we don't use this information we can obtain more that one outcome and we can't select the most preferred countermeasure to stop an `and`-attack.

Let us now to consider how to model this by using ASO programs.

*Example 2.* Consider the attack action depicted in left side of Figure 3, the attack action $x$ and the preference order over the corresponding countermeasures $a$, $b$, and $c$ generate the following ASO program $\langle \mathcal{P}_x, \Phi_x \rangle$:

$$\mathcal{P}_x \quad \mathtt{r_{x_1}} : \mathtt{x} \leftarrow, \quad \mathtt{r_{x_2}} : \mathtt{a} \vee \mathtt{b} \vee \mathtt{c} \leftarrow \mathtt{x} \qquad \Phi_x \quad \varrho_{\mathtt{x}} : \mathtt{a} > \mathtt{b} > \mathtt{c} \leftarrow \mathtt{x}$$

where the rule $r_{x_1}$ and $r_{x_2}$ introduce the action and the possible countermeasures, while $\varrho_x$ represents the preference order among them. The same result is obtained for the attack action $y$, the corresponding $\langle \mathcal{P}_y, \Phi_y \rangle$ program is:

$$\mathcal{P}_y \quad \mathtt{r_{y_1}} : \mathtt{y} \leftarrow, \mathtt{r_{y_2}} : \mathtt{a} \vee \mathtt{c} \leftarrow \mathtt{y} \qquad \Phi_y \quad \varrho_{\mathtt{y}} : \mathtt{c} > \mathtt{a} \leftarrow \mathtt{y}$$

In order to model the `and`-node *root*, a new program $\langle \mathcal{P}_{and}, \Phi_y, \Phi_x \rangle$ is generated combining the rules in $\langle \mathcal{P}_x, \Phi_x \rangle$ and $\langle \mathcal{P}_y, \Phi_y \rangle$ (see Figure 3). $\mathcal{P}_{and}$ introduces two new rules: $r_1$ represents the root action, while $r_2$ combines the action $x$ and $y$ in such way that only one of them must be stopped. The others rules are a added without any change. The answer sets of $\mathcal{P}_{and}$ are $M_1 = \{w, x, a\}$, $M_2 = \{w, x, b\}$, $M_3 = \{w, x, c\}$, $M_4 = \{w, y, c\}$ and $M_5 = \{w, y, a\}$. In order to establish the optimal answer set, the ASO semantics firstly constructs the satisfaction vectors $V_{M_1} = [\infty, 1]$[1], $V_{M_2} = [\infty, 2]$, $V_{M_3} = [\infty, 3]$, $V_{M_4} = [1, \infty]$ and $V_{M_5} = [2, \infty]$, reporting the satisfaction degree of

[1] In this application the irrelevance corresponds to the worst case.

each preference rule in the answer sets. Considering firstly $\varrho_y$ and then $\varrho_x$ the order among the answer set is $M_4 > M_5 > M_1 > M_2 > M_3$[2]. Concluding, the new order among the countermeasures is $c > a > b$. △

**or-composition.** The second method, called *or-composition*, was used to determine a preference order in the selection of the countermeasures able to stop an or-attack, i.e. an attack composed by a set of alternative actions one of which has to be successfully achieved to obtain the goal. The protection from this kind of attack consists in the protection from all the actions composing the or-attack. Intuitively, the most preferred strategy has to select the best countermeasure for each action or the countermeasure able to stop the bigger number of actions.

Again, in order to be able to select more than one countermeasure, a complete order among all of them need to be created. More formally, given the or-attack $X$, composed by $k$ actions $u_1, \ldots, u_k$, the sets of countermeasures $D_{u_1}, \ldots, D_{u_k}$ protecting $u_1, \ldots, u_k$ respectively, and the orders among countermeasure $(D_{u_i}, \succ_{u_i})$ for $i = \{1, \ldots, k\}$, then the or-composition is a new order $(D_X, \succ_X)$. The order is defined over the set $D_X$, whose elements $C_1, ..., C_n$ are the sets of countermeasures covering all the attacks $u_1, ...u_k$, and $\succ_X$ is defined as follows: the set $C$ is preferred to the set $C'$ if there is a permutation $\pi$ such that for all $i \in [1, \ldots, k]$, $c_i$ is not worst than $c'_{\pi(i)}$, i.e. for $k = k'$, $\exists\pi$ s.t. $c_i \not\prec c'_{\pi(i)}$. Notice also that when the same countermeasure is selected two times (to cover two different attacks), we consider its presence only one time.

Using the logic programming and the ASO semantics we can determine the preferred set of countermeasure faster than using the classical CP-net.

*Example 3.* As an example consider again the defence tree in the left side of Figure 3 and the or-attack $root'$ composed by three action $x$ and $y$. The corresponding ASO program $\langle \mathcal{P}_{or}, \Phi_y, \Phi_x \rangle$ is generated combing $\langle \mathcal{P}_x, \Phi_x \rangle$, $\langle \mathcal{P}_y, \Phi_y \rangle$ (see Figure 3). A new rule $r_1$, introduced in $\mathcal{P}_{or}$, represents $root'$, while the rules $r_2$, $r_3$ and $r_4$ model the or-attack, i.e. that all the three action must be stopped to stop the root.

The answer sets of $\mathcal{P}_{or}$ are $M_1 = \{root', x, y, a\}$ and $M_2 = \{root', x, y, c\}$[3] and describes the application of two alternative sets of countermeasures $\{a\}$ and $\{c\}$, protecting from the or-attack. In order to establish the optimal answer set, the ASO semantics firstly construct the satisfaction vectors $V_{M_1} = [1, 2]$ and $V_{M_2} = [2, 1]$. Then it compares these vectors, by firstly considering $\varrho_y \in \Phi_y$, obtaining that $V_{M_2} < V_{M_1}$. Concluding, $M_2$ is the optimal answer set and $\{c\}$ is the preferred set of countermeasures. △

*Implementation.* Given an IT system *root* and the corresponding CP-defence tree $\mathcal{T}$, the selection of the preferred defence strategy can be modelled by means of the corresponding logic program with preferences, then an ASO program

[2] Notice that both the answer set $M_1$ and $M_5$ contain countermeasure $a$. However, we collect the countermeasure to be applied starting from the best model, so from $M_4$ we collect $c$ from $M_5$ we collect $a$, and from $M_2$ we collect $b$.

[3] We reminded that the ASO semantics [8] only collect minimal answer set, so among the set $M_1 = \{root', x, y, a\}$ and $M'_1 = \{root', x, y, b, a\}$ only $M_1$ is considered because $M'_1$ is not minimal.

solver can be used to automatically obtain the set of the best countermeasure. We used for our security scenario analysis CPDT-SOLVER. CPDT-SOLVER is an application-oriented version of CHOPPER [9], realizing ASO semantics over the ranked answer set optimization program, under the assumption that $I \equiv \infty$.

The same scenarios can be represented also by using the CR-Prolog$_2$ semantics, the only difference is that the preference among the countermeasures and the corresponding ordering can be written by using only one rule. An implementation is available for download from `http://www.krlab.cs.ttu.edu/Software/`.

## 5 Conclusion

In this paper we use the ASO semantics as an instrument to represent and solve the problem of countermeasure selection in a security scenario defined using a defence tree and a CP-net. The CP-net is used to describe the dependency among attacks and countermeasures, the preferences among the countermeasures, and (possibly) the order among attacks (depending from their seriousness or other parameters), whilst the defense tree represent the scenario associating each countermeasure to each attack action. The structure of the defence tree is built bottom-up by connecting with `and` nodes actions to be performed at a whole and with `or` node actions that by themselves can lead to a successful attack. In particular, the composition of CP-net needed to deal with `and`/`or` nodes is substituted by a composition of the corresponding ASO program. The composition of the ASO programs results extremely easy w.r.t. the composition of CP-net, and automatically remove cycles that can be obtained with `and`-composition.

## References

1. Balduccini, M., Gelfond, M.: Logic programs with consistency-restoring rules. In: AAAI 2003. Int. Symp. on Logical Formalization of Commonsense Reasoning. Spring Symposium Series, pp. 9–18 (2003)
2. Balduccini, M., Mellarkod, V.S.: Cr-prolog2: Cr-prolog with ordered disjunction. In: ASP03 Answer Set Programming, vol. 78 (2003)
3. Bistarelli, S., Fioravanti, F., Peretti, P.: Defense tree for economic evaluations of security investment. In: 1st Int. Conf. on Availability, Reliability and Security (ARES'06), pp. 416–423 (2006)
4. Bistarelli, S., Fioravanti, F., Peretti, P.: Using cp-nets as a guide for countermeasure selection. In: ACM SAC2007. ACM Press, New York (2007)
5. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H., Poole, D.: Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. JAIR 21 (2004)
6. Brafman, R.I., Dimopolous, Y.: Extended semantics and optimization algorithms for cp-networks. Computational Intelligence 20(2), 218–245 (2004)
7. Brewka, G.: Logic programming with ordered disjunction. In: 18th Conf. on Artificial intelligence, pp. 100–105. American Association for AI (2002)
8. Brewka, G., Niemela, I., Truszczynski, M.: Answer set optimization. In: Proc. of the 18th Int. Joint Conf. on Artificial Intelligence, pp. 867–872 (2003)

9. Caroprese, L., Trubitsyna, I., Zumpano, E.: Implementing prioritized reasoning in logic programming. In: Proc. of the ICEIS (2007)
10. Delgrande, J.P., Schaub, T., Tompits, H., Wang, K.: A classification and survey of preference handling approaches in nonmonotonic reasoning. Computational Intelligence 20(2), 308–334 (2004)
11. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databasesg. New Generation Computing 9, 365–385 (1991)
12. Schneier, B.: Attack trees: Modeling security threats. Dr. Dobb's Journal (1999)