Report Number: WUCS-92-46

1992-11-01

Computing Specificity

Authors: Ronald Loui, J. Norman, K. Stiefvater, A. Merrill, A. Costello, and J. Olson

This note reports on an effort to implement a version of Poole's rule for specificity. Relatively, efficient implementation relies on correcting and improving a pruning lemma of Simari-Loui [92]. This in turn requires revision of Poole's specificity concept.
The resulting system is a usable knowledge representation system with first-order-language and defeasible reasoning. Sample input and output are included in an appendix. It is a good candidate for multiple inheritance applications; it is useful for planning, but limited by the underlying search for plans.

Computing Specificity

R. Loui, J. Norman, K. Stiefvater, A. Merrill, A. Costello and J. Olson

WUCS-92-46

November 1992

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130-4899

Computing Specificity

R. Loui[1]
J. Norman[2]
K. Stiefvater[3]
A. Merrill[4]
A. Costello[5]
J. Olson[6]
Department of Computer Science
Washington University
St. Louis

Abstract. This note reports on an effort to implement a version of Poole's rule for specificity. Relatively efficient implementation relies on correcting and improving a pruning lemma of Simari-Loui [92]. This in turn requires revision of Poole's specificity concept.

The resulting system is a usable knowledge representation system with first-order-language and defeasible reasoning. Sample input and output are included in an appendix. It is a good candidate for multiple inheritance applications; it is useful for planning, but limited by the underlying search for plans.

1

Much reasoning proceeds by argument. Automating this kind of reasoning requires theories about how to adjudicate disputes involving competing arguments. In recent work this has been described as determining a preference among arguments. A syntactic criterion for preference, especially one based on a concept of defeat among arguments, would be of practical use.

After numerous attempts at identifying a suitable convention that determines when one non-demonstrative argument defeats another, two conclusions can be drawn: (1) there may be many suitable conventions (though this should not necessarily be described as a "clash of intuitions"); and (2) in cases where the convention is not fixed (either by the field as a whole, or by a representer of knowledge, or by agreement among the conflicting parties to a dispute), superiority of one argument over another should be determined by meta-argument

[1]Supported by NSF R-9008012.

[2]Current address: Seyfarth, Shaw, Fairweather, and Geraldson, 55 E. Monroe, Chicago, IL; also fellow of The Center for Intelligent Computer Systems, Washington University.

[3]Supported by NSF CDA-9102090.
[4]Supported by NSF CDA-9123643.
[5]Supported by NSF R-9008012 and CDA-9102090.
[6]Supported by NSF R-57135A.

during the (object-level) dispute.

The latter alternative has not yet been fully explored. Presumably it would require drawing analogies to benchmark disputes, or concocting a set of non-demonstrative reasons for claiming one argument is better than another. Actual arguments among researchers about which convention to adopt proceed by the former, but off-line (without reference to a particular dispute). The early investigation of Loui [87] does the latter.

Though the field may lack the courage to forge a convention, and may be willing to retreat to on-line (mid-dispute) meta-argument, further study of promising conventions is appropriate. Even if parties to a dispute or a knowledge representer were willing simply to adopt an imperfect convention, a syntactic criterion for argument preference, there are still not many good candidates that could be adopted. There are a few suitable inheritance systems (e.g., Horty-Thomason-Touretzky [90a,90b], Nute [90], Stein [90], Guha [90], Padgham [89]), but the number of first-order systems is small (e.g., Simari-Loui [92], Geffner-Pearl [92], Pollock [92]). Note that we are restricting the discussion to systems that provide a syntactic criterion for preference; hence, they do not require the user to supply ordering information explicitly (e.g., Ginsberg [88], Baker-Ginsberg [89], Konolige [88], Lin-Shoham [89], Eklund [91], etc.).

A major factor is computation. The systems proposed appear expensive to implement; the queries to theses systems appear expensive to compute. Most computations will call a theorem-prover as a subroutine, providing a bound on computation for negation as failure. Most call the theorem-prover a combinatorial number of times.

We cannot speak of ultimate tractability for these systems, since the underlying logic is semi-decidable. However, since the theorem-proving is a constant computation in practice, it is reasonable to ask what can be done about the number of times this computation is invoked (see Kautz-Selman [88,91], Stillman [90] for complexity results regarding inheritance, the less expressive kind of system with syntactic preference criterion).

Systems with implicit preference are not often implemented due to a lack of interest in this last question (as many authors have recently noted; see the AAAI Spring Symposium on Implemented Knowledge Representation Systems [91]). This is especially unfortunate since the choice of a convention is informed by experience with using the convention.

2

One of the syntactic specifications of preference that continues to be the most promising is due to Poole [85]. Simari-Loui [90,92] is an effort to make Poole's rule usable. In particular, they embed Poole's rule in Pollock's theory of defeasible reasoning [87,90,92], thus providing an implicit preference, specificity, where Pollock requires more explicit intervention.

Poole's rule holds one argument, which he calls a theory, to be more specific than another if there is some way of activating the first without activating the second; and not vice versa. Note the asymmetry occurs both outside and inside the existential quantification (actually the outside is technically an antisymmetry; the inside is asymmetry figuratively). Pollock's theory provides a way of applying specificity to an interconnected web of arguments each of which might attack another's subargument. Pollock's own system does not allow implicit preference of arguments.

As noted as Simari-Loui [92] went to press however, the computationally most important lemma of the article is wrong. It states that checking specificity is equivalent to checking that the antecedents of the less specific theory can be derived from the antecedents of the more specific theory. This is not even true for the system restricted to Horn clause-like rules as implemented in Prolog by the authors.

Getting this lemma right is crucial because computing Poole's specificity as implied by the definitions is impossible. This is the issue that McDermott raised in [87]:

> If this can really be made to work, then it is the only direct rebuttal .... However, it is hard to tell ...exactly how Poole's idea is to work. ...In the definitions ..., proving part (a) would require looking for an arbitrary subset of facts .... Proving part (b) is even worse ....

and it is one reason why David Poole's THEORIST [88] does not attempt to order theories according to Poole's own rule (see also Prakken's implementational concerns in [91] in an attempt to apply the system to simple legal reasoning).

The lemma states that a theory T_1 is more specific than T_2 just in case for every x, an antecedent of a rule used in T_2, x can be defeasibly derived from K_N, the necessary evidence, T_2's rules, and the antecedents of T_1's rules:

$$(\forall x \in An(T_2))(K_N \cup An(T_1) \cup T_2 \mid\sim x).$$

In the case of

$$< \{Penguin(O) \ >\!\!- \ \neg Flies(O)\}, \neg Flies(O) >$$

versus

$$< \{Bird(O) \ >\!\!- \ Flies(O)\}, Flies(O) >,$$

for example, $Bird(O)$ can be defeasibly derived from $Penguin(O)$. And in

$$< \{Cat(G) \ >\!\!- \ Aloof(G), \\ Aloof(G) \ >\!\!- \ \neg LikesPeople(G)\}, \neg LikesPeople(G) >$$

versus

$$< \{Cat(G) \ >\!\!- \ LikesPeople(G)\}, LikesPeople(G) >,$$

$Cat(G)$ defeasibly derives both $Cat(G)$ and $Aloof(G)$.

This last example is a simple counterexample to the lemma. If the lemma were right, then not only is the theory for $LikesPeople(G)$ more specific than the theory for $\neg LikesPeople(G)$ which we have just said is desirable, but also vice versa, which is not desirable. Not only is this intuitively undesirable, but it also violates the antisymmetry of specificity. The required relationship between antecedents is necessary for specificity, but not sufficient.[7]

Correcting this error requires[8] first that the definition of specificity be repaired. As it stands, it is vulnerable to some counter-intuitive behavior. This counter-intuitive behavior plagues Poole's rule generally, not just our use of it.

The argument, for example,

$$\begin{aligned} A \;&\multimap\; B \wedge C \\ B \;&\multimap\; D \\ C \;&\multimap\; E \end{aligned}$$

should be more specific than

$$\begin{aligned} \neg A \;&\multimap\; B \\ B \;&\multimap\; D. \end{aligned}$$

But it is not more specific according to the unadulterated Poole definition, because of two separate flaws in the definition.

The first reason is that $E \wedge (B \vee \neg C)$ allows the former theory to be activated without activating the latter. This prevents the desired conclusion that the former theory is more specific. What is basically wrong here is that the weaker theory should be allowed to use the defeasible rule $E \;\mathrel{>\!\!\!-}\; C$, in order to derive (defeasibly) B.

To see the second flaw, consider $E \wedge (A \vee \neg C)$, which is also disjunctive, but this time uses the disjunction to derive the theory's ultimate conclusion. This is essentially a side-stepping of the non-triviality condition for activators. The non-triviality condition should be strengthened.

---

[7] Where the alleged proof goes wrong is quite easy to see. When there is specificity, every antecedent of the weaker theory can be derived from every antecedent of the stronger theory, but not necessarily from an activator of the stronger theory (an activator is a sentence of the proscribed contingent kinds, which allows a theory's conclusion to be derived, using the necessary evidence, K_N, and the theory's rules). That is, it may be possible to activate the theory without allowing *all* of its antecedents to be defeasibly derived, which is just plain to see.

Henry Prakken [92] has noticed that Theorem 4.16 of the paper is also in error. An example can be given in which two arguments defeat each other. Specificity is antisymmetric, but not defeat. The proof goes sour at "the same is true for defeat."

[8] Prakken [92] corrects the error in an equally intuitive way. He suggests restricting the search for activators to those that activate theories in eactly the same way that the evidence does (this must be done carefully if there are multiple derivations). This fix for Poole also appears to allow a pruning lemma; however, it would involve cutsets of derivation trees, which are combinatorial in number).

Fixing the flaws in Poole's rule is important because this kind of comparison is the comparison used in the Yale Shooting Problem arguments (Hanks-McDermott [87]), as exhibited among the examples in the paper by Simari-Loui.

The rule also suffers in an example reminiscent of Royal Elephants (Sandewall [86]): Consider

$$\begin{aligned} D &\prec B \wedge C \\ B \wedge C &\prec A \end{aligned}$$

compared with what should be an inferior theory:

$$\begin{aligned} \neg D &\prec B \\ B &\prec E. \end{aligned}$$

Neither is more specific by Poole's rule. The example appears to require right-weakening of rules: allowing rules to be derived from rules by weakening the consequent.[9] But right-weakening is notoriously problematic.[10]

The proposed development for specificity, which fixes both kinds of counterintuitive behavior due to disjunction, which allows proper treatment of the last example, and which allows a proper pruning lemma, is as follows.

Let $\Delta^{\downarrow} \subseteq L^2$, $K_N \subseteq L$, and $K_C \subseteq L$ be, respectively, instantiated defeasible rules, necessary (background) knowledge, and contingent knowledge (evidence), referring to a first-order language L.

---

[9] For example,

If $A \succ B \wedge C$, then $A \succ C$;

then the theory

$$\begin{aligned} D &\prec B \wedge C \\ B &\prec E \\ C &\prec A \end{aligned}$$

would be the defeater of the weaker theory. The first theory does not defeat the third, but with right-weakening, it allows the construction of a third theory which directly accounts for the considerations used in the weaker theory, and which ought to be more specific.

[10] When there are competing arguments, such as

$$B \prec A$$

versus

$$\neg B \prec A,$$

a rule can be right-weakened with an arbitrary dilution, such as

$A \succ B$, therefore, $A \succ B \vee C$,

allowing an argument for C. This cannot be done without right-weakening, since arguments must have consistent intermediate claims. This particular case is not so bad, since the argument

$$\begin{aligned} C &\prec (B \vee C) \wedge \neg B \\ B \vee C &\prec A \\ \neg B &\prec A \end{aligned}$$

has counterargument

$$B \prec A,$$

but if $A \wedge D \succ \neg B$, the arbitrary dilution can actually be supported. Instead, fix the rule for specificity so that it treats the problem properly without requiring right-weakening.

An argument, $< T, h >$ has $T \subseteq \Delta^{\downarrow}$ and h derivable from $T \cup K_N \cup K_C$, where derivation may use rules of FOL, and a modus ponens for defeasible rules, i.e.

$$\frac{\vdash p}{\mid\sim p}$$

$$\frac{\mid\sim p, \quad p \succ\!\!- q}{\mid\sim q.}$$

Also, T is minimal; no proper subset of T allows derivation of h.

A rule, R, is a *top rule* of argument $< T, h >$ just in case its consequent, $Con(R)$, is not needed for the derivation of anything but the argument's conclusion. Because the rules used in arguments are a minimal set, that is equivalent to saying that the antecedent of any rule can be derived (from evidence) using rules other than this top rule.[11]

Definition. $\mathbf{Top}(R, < T, h >)$ iff for every r in T, $An(r)$ can be defeasibly derived from $K_N \cup K_C$ using $T - \{R\}$.

Example. $B \succ\!\!- C$ is a top rule in the argument from A to C, using $A \succ\!\!- B$, and $B \succ\!\!- C$.

Let Δ be a set of defeasible rules, let h be a sentence in the language, L, and let A be a set of sentences in L.

A finite sequence of sentences, $< B_1, \ldots, B_n >$ is a *consistent defeasible derivation (CD-derivation)* of h from A using rules Δ just in case h is derived by A's activating ground instances of rules, and the set of all intermediate sentences is consistent in L.

Definition. $< B_1, \ldots, B_n >$ is a **CD-derivation of h from A** iff

1. $B_n = h$;
2. For each B_i, either

   a. $\{B_j \mid j < i\} \cup A \vdash B_i$; or

   b. for some ground instance of a rule R in Δ,
   $An(R) = B_j$ for some $j < i$
   and $B_i = Con(R)$;

3. $\{B_i \mid i \leq n\} \nvdash \bot$.

[11] It is not sufficient to say that a rule is top just in case it participates in eliminating a literal from the goal clause: consider $< \{Q \succ\!\!- R, S \succ\!\!- T\}, R \wedge T >$ which is an argument for $R \wedge T$; only the latter is a top rule; otherwise, the argument would not defeat $< \{Q \succ\!\!- \neg R\}, \neg R >$, which it should.

Example (continued). $< A, B, C >$ is a CD-derivation of C from A using $A \succ\!\!- B$ and $B \succ\!\!- C$.

Definition. There is a **CD-derivation** of h from A **using** Δ **with all top rules** of $< T, h >$ just in case

1. there is a CD-derivation of h from A using Δ;

2. for each x such that $Top(x, < T, h >)$, there is no CD-derivation of h from A using $\Delta - \{x\}$.

$< T_1, h_1 >$ is more specific than $< T_2, h_2 >$ just in case some legitimate sentence activates T_2 for h_2 without activating T_1 for h_1, using CD-derivations from *the two theories' combined set of rules*, using *every top rule* of T_2; and there is no such *asymmetric activator* of T_1 for h_1 that does not also activate T_2 for h_2.

The requirement to use top rules is just a strengthening of Poole's non-triviality condition that asymmetric activators do not activate the theory simply by FOL rules, side-stepping the defeasible rules. The combination of theories is more profound. It signals the importance of pairwise comparison as opposed to an n-wise, holistic evaluation of merit (which is what Geffner-Pearl tends toward) on one extreme, or a conception of specificity as intrinsic, perhaps even measurable (which is what the algebra of Simari-Loui suggests), on the other extreme. Transitivity no longer holds of specificity.

That is,

Definition. e is an **asymmetric activator** of $< T_1, h_1 >$ **but not** $< T_2, h_2 >$ just in case

1. there is some CD-derivation of h_1 from $K_N \cup \{e\}$ using $T_1 \cup T_2$ with all top rules of $< T_1, h_1 >$;

2. there is no CD-derivation of h_2 from $K_N \cup \{e\}$ using $T_1 \cup T_2$ with all top rules of $< T_2, h_2 >$.

Let eAA_{inotj} symbolize that e is an asymmetric activator of $< T_i, h_i >$ but not $< T_j, h_j >$.

Definition. $< T_1, h_1 >$ is **more specific** than $< T_2, h_2 >$ just in case

1. there is some asymmetric activator in S_C of $< T_2, h_2 >$ but not of $< T_1, h_1 >$;
i.e., there is some $e \in S_C$ s.t. eAA_{2not1}; and

2. there is no asymmetric activator in S_C of $< T_1, h_1 >$ but not $< T_2, h_2 >$;
i.e., there is no $e \in S_C$ s.t. eAA_{1not2}.

Given $< T_1, h_1 >$ and $< T_2, h_2 >$, use the symbolization $e \ \vdash\!\!\rightarrow_{T_i} f$ to assert the existence of a CD-derivation of f from $K_N \cup e$ using $T_1 \cup T_2$ with all top rules of $< T_i, h_i >$. $e \ \vdash\!\!\rightarrow \ f$ if there is a CD-derivation at all from $K_N \cup e$ using $T_1 \cup T_2$, not requiring use of top rules. Note that $\vdash\!\!\rightarrow$ and $\vdash\!\!\rightarrow_{T_i}$ are defined only for a pair of theories being compared.

Note also that

eAA_{1not2} just in case

1. $e \ \vdash\!\!\rightarrow_{T_1} h_1$;

2. $e \ \not\vdash\!\!\rightarrow_{T_2} h_2$;

The new pruning lemma makes use of both the top rule restriction and the union of theories when checking activation.

To find whether there is an asymmetric activator of $< T_1, h_1 >$ but not $< T_2, h_2 >$ it is usually sufficient to check whether the conjoined antecedents of the top rules of $< T_1, h_1 >$ is an asymmetric activator of $< T_1, h_1 >$ but not $< T_2, h_2 >$. For simplicity, first assume that h_1 can be derived from the conjoined consequents of top rules in $< T_1, h_1 >$, the last step in deriving the theory's conclusion uses just K_N and the consequents of top rules; that is, intermediate conclusions from consequents of non-top rules are used only to derive antecedents of later rules.

Lemma (restricted pruning). For any arguments $< T_1, h_1 >$ and $< T_2, h_2 >$, there exists an asymmetric activator of $< T_1, h_1 >$ but not $< T_2, h_2 >$ just in case the following has the property AA_{1not2}:
$Conjoin_\Gamma(An(R_i))$,
where $\Gamma = \{R_i : Top(R_i, < T_1, h_1 >)\}$;
under the assumption that h_1 can be derived from the conjoined consequents of top rules in $< T_1, h_1 >$, i.e.,
$Conjoin_\Gamma(Con(R_i)) \ \vdash\!\!\rightarrow \ h_1$.

Proof.

1. First consider the case where $< T_1, h_1 >$ has a single top rule.
Suppose there is an e such that eAA_{1not2}. That is, $e \ \vdash\!\!\rightarrow_{T_1} h_1$ and $e \ \not\vdash\!\!\rightarrow_{T_2} h_2$. Let $e' = An(R)$, where by assumption, R is the only rule in T_1 such that $Top(R, < T_1, h_1 >)$.
Clearly, $e' \ \vdash\!\!\rightarrow_{T_1} h_1$, since we assume $Conjoin_\Gamma(Con(R_i)) \ \vdash\!\!\rightarrow \ h_1$. So e' is also an asymmetric activator if $e' \ \not\vdash\!\!\rightarrow_{T_2} h_2$.

Assume to the contrary that $e' \vdash\!\!\!\rightarrow_{T_2} h_2$. Recall that $e \vdash\!\!\!\rightarrow_{T_1} h_1$, so $e \vdash\!\!\!\rightarrow e'$, since e' is just the antecedent of the top rule which must be used. Chain this with the assumption that $e' \vdash\!\!\!\rightarrow_{T_2} h_2$, and get $e \vdash\!\!\!\rightarrow_{T_2} h_2$. But e is supposed to be an asymmetric activator of $< T_1, h_1 >$ but not $< T_2, h_2 >$. This is a contradiction. So e' must be an asymmetric activator.

2. Next, consider the case where $< T_1, h_1 >$has multiple top rules. Let e' be $Conjoin_\Gamma(An(R_i))$. The same argument applies, but it is no longer obvious that $e \vdash\!\!\!\rightarrow e'$.

Assume $e \vdash\!\!\!\rightarrow_{T_1} h_1$. Consider any rule, $R \in \Gamma$, i.e., any R such that $Top(R, < T_1, h_1 >)$. Show that $e \vdash\!\!\!\rightarrow An(R)$. This suffices to show that $e \vdash\!\!\!\rightarrow e'$. Assume to the contrary that $e \not\vdash\!\!\!\rightarrow An(R)$. This is a contradiction, because $e \vdash\!\!\!\rightarrow_{T_1} h_1$ requires that any CD-derivation of h_1 from $K_N \cup e$ using $T_1 \cup T_2$ use all of the top rules of $< T_1, h_1 >$, including R, and it is impossible to use R without deriving $An(R)$.

Now relax the assumption regarding the derivability of h_1 from consequents of top rules. The full lemma is that whenever there is an asymmetric activator of $< T_1, h_1 >$ but not $< T_2, h_2 >$, an asymmetric activator is formed by conjoining (1) the antecedents of top rules with (2) the conditional whose antecedent (2a) conjoins the consequents of non-top rules, and whose consequent (2b) is the claim supported by the argument.

Lemma. For any arguments $< T_1, h_1 >$ and $< T_2, h_2 >$, there exists an asymmetric activator of $< T_1, h_1 >$ but not $< T_2, h_2 >$ just in case the following has the property AA_{1not2}:
$Conjoin_\Gamma(An(R_i))$

$\wedge$

$(Conjoin_\Gamma(Con(R_i)) \supset h_1)$.

where Γ is as before.

Proof.

The proof again begins: suppose some eAA_{1not2}; i.e., $e \vdash\!\!\!\rightarrow_{T_1} h_1$, $e \not\vdash\!\!\!\rightarrow_{T_2} h_2$. e' is the conjunction of top rule antecedents, and $e''$is e' conjoined with the material conditional as above. We want $e''AA_{1not2}$. Clearly $e'' \vdash\!\!\!\rightarrow_{T_1} h_1$. Show $e'' \not\vdash\!\!\!\rightarrow_{T_2} h_2$. Suppose

it did, i.e., $e'' \vdash\!\!\!\rightarrow_{T_2} h_2$: $e \vdash\!\!\!\rightarrow_{T_1} h_1$, so $e \vdash\!\!\!\rightarrow e''$ since all top rules must be used and e'' is the weakest sentence that activates all top rules and also derives h_1. This property of being weakest is the key observation. Chaining, $e \vdash\!\!\!\rightarrow_{T_2} h_2$, but this is a contradiction. So $e'' A A_{1not2}$ if for any e, $e A A_{1not2}$.

3

The revised rules have been implemented twice. A version with an underlying first-order logic, on which this section focuses, was implemented in C and is quick; it is primarily limited by the underlying resolution theorem-prover.[12] The second version is in LISP with a restricted propositional language (just negation of atomic formulae and conjunction), with provision for analogical (case-based) reasoning, and with additional features peculiar to certain forms of legal reasoning (a complete report is forthcoming, Loui-Norman [92]).[13]

Detailed examples of the input and output of the C program are included in an appendix.

A set-of-support resolution theorem-prover is used to construct arguments. Two major modifications are required.

First, defeasible rules do not contrapose (this is a disputed issue; we fail to contrapose primarily because we view defeasible rules as policies). This restriction is enforced by ordering clauses and requiring that a descendant of the goal clause resolve with the consequent-clause of a defeasible rule. In order to allow disjunction and conjunction in antecedent and consequent, every defeasible rule is replaced with a "fake" rule, e.g., $fake_1 \succ\!\!- fake_2$, with $fake_1$ being biconditionally equivalent to the antecedent, and $fake_2$ biconditionally equivalent to the consequent.

Second, reasoning by cases is permitted for FOL rules, but not for defeasible rules (this is to prevent Simpson's paradox problems with defeasible rules, as reported in Loui [87], Pearl [88], Neufeld-Poole [88], Neufeld [89] and others). This restriction is enforced by allowing reasoning by cases to occur only within a *family*, where every use of a defeasible rule heads a new family.

The theorem-prover is called at all times with a bound on number of resolution attempts and a bound on successes. The computation times indicated reflect many current inefficiencies; the most costly are the practice of generating multiple derivations for a single theory, hence, redundant clauses; the lack of the

[12] The main programmers were Adam Costello, Andrew Merrill, and Ronald Loui; the 92k, 3300 lines of source code contain a dedication to the late computer scientist, Eugene Nathan Johnson, c. 1944 – 1984; the program is called "nathan".

[13] The main programmer was Jon Olson; its 52k, 2000 lines of source code are called "lmnop" after the initials of the last names of its designers.

usual indexing of clauses by predicate names occurring within them; and, the current bounds of 1000 resolvents and 15000 attempts, for every proof failure.

The system currently does not check that arguments are consistent with evidence. Instead, it allows them as arguments that are easily defeated by "arguments" that are just evidence. It also does not check that arguments are minimal, and an example can be given where this matters.

Nute [86], Poole [88], Baker-Ginsberg [91], and Pollock [92] are the most related efforts. A full comparison is beyond the scope of this work. However, brief remarks are in order.

Nute's system compares just the specificity of antecedents of rules. In contrast, we compare the derivations (the arguments), each derivation using potentially many rules. As mentioned earlier, Poole's THEORIST does not attempt to determine specificity, even though it computes arguments in much the same framework.

Baker-Ginsberg is based on multi-valued logics.[14] Ginsberg's [88] idea was to provide not just {*true, false, unknown, both*} as the possible valences of sentences, but to augment the set to {*true, false, defeasibly true, defeasibly false, unknown, both*} for default logics, and {*true, false, defeasibly true 1, defeasibly false 1, both 1, defeasibly true 2, defeasibly false 2, both 2, unknown, both*} for 2-level prioritized default logics. The more levels of priority required, the more valences introduced. This requires explicit attachment of priorities to default rules. We would assert

$$bird(x) \;\succ\!\!-\; flies(x) \text{ and}$$
$$penguin(x) \;\succ\!\!-\; \neg flies(x),$$

and determine a preference syntactically. In contrast, Baker-Ginsberg write

"Flies(x) :- Bird(x) P3 (priority 1)"
"Not(Flies(x)) :- Penguin(x) P4 (priority 2)"

Pollock [92] is an elegant treatment of defeasibilty where all defeat relations are explicit. The underlying theorem prover is natural deduction, which obviates the need for families and ordered rules, as encountered in our design. However, it is unclear how focus and backward chaining work in Pollock's OSCAR system; our set-of-support resolution is more familiar to AI audiences. More importantly, where we would assert

$$bird(x) \;\succ\!\!-\; flies(x) \text{ and}$$
$$penguin(x) \;\succ\!\!-\; \neg flies(x),$$

Pollock writes

[14] It is unknown whether Baker and Ginsberg's underlying logic is fully first-order, or whether, as in the examples given, predicates are restricted to one variable.

"(bird x) ||⇒ (flies x)",
"(penguin x) ||⇒ (not (flies x))", and
"(penguin x) ||⇒ ((bird x) ⊗ (flies x))".

All of our implicit specificity must be made explicit in this manner.

In comparison with the implementation in Simari-Loui [92], the underlying language is first-order, not restricted to Horn clauses. Also, a different pruning lemma is used, as detailed.

4

We correct the Poole-based part of the system proposed in Simari-Loui.[15] The corrections are unfortunate considering the original authors' hope that the rules

[15]Touretzky, Horty, and Thomason have attacked the Pollock-based part of the system [91]. We are sympathetic to their concerns but feel no revision is required. They hold that reinstatement is inappropriate when the original argument, its counterargument, and the putative reinstater all contend the same proposition. They have no dispute with reinstatement that occurs when the reinstater attacks a subargument of the counterargument. At issue is a set of rules such as:

birds tend to fly;

chickens tend not to fly;

wild chickens tend to fly.

There is no dispute that a wild chicken should fly, but it is unusual in this example to say the argument that a wild chicken flies reinstates the argument that it flies in virtue of being a bird, by eliminating its only counterargument, that it does not fly in virtue of being a chicken. Clearly

birds tend to fly-because-they-are-birds

will not be reinstated by

wild chickens tend to fly-because-they-are-wild-chickens.

Part of the oddness of reinstatement here is that the reasons on which arguments are based imply explanations; it is difficult to regard them simply as the basis for inference. But even restricting attention to the idea that an argument is reinstated if its counterarguments are defeated, which makes no claims about explanations, the wild chicken example gives pause. Wildness has nothing to do with the reason chickens are exceptional birds.

Consider on the other hand,

Porsches tend to be fun;

Porsches co-developed with VW are not fun;

refined Porsches co-developed with VW are fun.

Refined co-developed Porsches are fun because they're Porsches, not because they are refined. There are statistical examples of both kinds: a team that usually wins, but loses at night, wins at night in well-lit stadia. In this case, there is reinstatement. But if it wins at night with a particular pitcher, this apparently does not address the reason why the subclass is exceptional. William Chen offers this example:

animals are uncivilized;

people are civilized;

murderers are uncivilized.

and their form would exhibit more permanence than those of competing systems. Poole's rule seemed to be a tidy replacement for the potentially numerous meta-reasons for superiority of one argument over another (for example, as advocated by Loui [87], and more deeply voiced by Konolige and Pollack [89]). Poole's rule has largely escaped pointed criticism of its behavior. However, its appeal was largely based on its tidiness, not on its obviousness (we still have seen no exact text of Popper, though Poole makes the attribution). The merit of Poole's idea may depend on the adequacy of repairs such as this.

However, the development of actual systems resulting from longstanding interest in a single clear idea can no doubt influence the ultimate choice of convention.

5 References

Special Issue on Implemented Knowledge Representation and Reasoning Systems, SIGART Bulletin 2, 1991.

Baker, A. and M. Ginbsberg. "A theorem prover for prioritized circumscription," *Proc. IJCAI*, 1989.

Brewka, G. "Preferred subtheories," *Proc. IJCAI*, 1989.

Eklund, P. "An epistemic approach to interactive design in multiple inheritance hierarchies," dissertation, Linkoeping University, 1991.

Geffner, H. and J. Pearl. "Conditional entailment: bridging two approaches to default reasoning," *Artificial Intelligence 53*, 1992.

Gelfond, M., Lifschitz, V., Przymusinska, H., and Truszczynski, M. "Disjunctive defaults," *Proc. KR*, 1991.

There is reinstatement, because murderers aren't people; they're animals!

Mitigating the implicit concerns of Touretzky, Horty, and Thomason, a system is usable by the wary knowledge representer whether it has reinstatement or not. If the system reinstates automatically, then the following set of rules prevents unwanted reinstatement:

> birds are things-that-fly-because-they-are-birds, which tend to fly;
>
> chickens tend not to fly;
>
> wild chickens tend to fly, but are not things-that-fly-because-they-are-birds.

If, on the other hand, the system does not automatically reinstate, reinstatement can be enabled with the following representation:

> Porsches are fun;
>
> Porsches co-developed with VW are the-kind-of-thing-that-is-no-fun, which are things that tend not to be fun;
>
> refined Porsches co-developed with VW are fun, and are not the kind-of-thing-that-is-no-fun.

Touretzky, Thomason, and Horty are once again pointing to a mere clash of intuitions. We see no need to revise the Pollock-based part of the system in response to their essay.

Ginsberg, M. "Multivalued logics: a uniform approach to reasoning in artificial intelligence," *Computational Intelligence 4*, 1988.

Grosof, B. "Generalizing prioritization," *Proc. KR*, 1991.

Guha, R. "The representation of defaults in Cyc," *Proc. AAAI*, 1990.

Hanks, S. and D. McDermott. "Nonmonotonic logic and temporal projection," *Artificial Intelligence 33*, 1987.

Horty, J. , Thomason, R., and Touretzky, D. "A skeptical theory of inheritance in nonmonotonic semantic nets," *Artificial Intelligence 42*, 1990a.

Horty, J. , Thomason, R., and Touretzky, D. "Boolean extensions of inheritance networks," *Proc. AAAI*, 1990b.

Kautz, H. and B. Selman. "Complexity of model-preference default theories," *Workshop on Nonmonotonic Reasoning*, 1988.

Kautz, H. and B. Selman. "Hard problems for simple default logics," *Artificial Intelligence 49*, 1991.

Konolige, K. "Hierarchic autoepistemic theories for nonmonotonic reasoning," *Proc. AAAI*, 1988.

Konolige, K. and M. Pollack. "Ascribing plans to agents," *Proc. IJCAI*, 1989.

Lifschitz, V. "Computing circumscription," Proc. IJCAI, 1985.

Lin, F. and Y. Shoham. "Argument systems," Stanford Computer Science technical report 89-1243, 1989.

Lin, F. "Formalizing various intuitions about inheritance in logic programs," *Workshop on Nonmonotonic Reasoning*, 1990.

Loui, R. "Theory and computation of uncertain inference and decision," dissertation, University of Rochester, 1987.

Loui, R. "Defeat among arguments: a system of defeasible inference," *Computational Intelligence 3*, 1987.

Loui, R. "Process and Policy," submitted to *Cognitive Science*, 1992.

Loui, R. and J. Norman. "Design of a case-based and rule-based legal reasoner with some automatic comparison of arguments," in preparation, 1992.

McDermott, D. "AI, logic, and the frame problem," in *The Frame Problem in AI*, Frank Brown, ed., pp. 105-118, Morgan Kaufman, 1987.

Neufeld, E. and D. Poole. "Probabilistic semantics and defaults," Proc. Uncertainty in AI, 1988.

Neufeld, E. "Defaults and probabilities; extensions and coherence," Proc. KR, 1989.

Nute, D. "LDR: a logic for defeasible reasoning," University of Georgia Advanced Computational Methods Center, Report 01-0013, 1986.

Nute, D. "Defeasible logic and the frame problem," in *Knowledge Representation and Defeasible Reasoning*, H. Kyburg, R. Loui, and G. Carlson, eds. Kluwer, 1990.

Padgham, L. "Nonmonotonic inheritance for an object-oriented knowledge base," dissertation, Linkoeping University, 1989.

Pearl, J. *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufman, 1988.

Pollock, J. "Defeasible reasoning," *Cognitive Science 11*, 1987.

Pollock, J. *Nomic Probability and The Foundations of Induction*, Oxford, 1990.

Pollock, J. "How to reason defeasibly," *Artificial Intelligence 57*, 1992.

Poole, D. "On the comparison of theories: preferring the most specific explanation," *Proc. IJCAI*, 1985.

Poole, D. "A logical framework for default reasoning," *Artificial Intelligence 36*, 1988.

Poole, D. "What the lottery paradox tells us about default reasoning," *Proc. KR*, 1989.

Poole, D. "Dialectics and specificity" *Workshop on Nonmonotonic Reasoning*, 1990.

Prakken, H. "A tool in modelling disagreement in law: preferring the most specific argument," *Proc. of the Third Conference on AI and Law*, ACM Press, 1991.

Prakken, H. "An argumentation framework in default logic," working paper, Computer/Law Institute, Free University Amsterdam, 1992.

Sandewall, E. "Non-monotonic inference rules for multiple inheritance with exceptions," *Proc. IEEE 74*, 1986.

Shoham, Y. "Nonmonotonic logics: meaning and utility," Proc. IJCAI, 1987.

Simari, G. and R. Loui "Confluence of argument systems: Poole's rules revisited" *Workshop on Nonmonotonic Reasoning*, 1990.

Simari, G. and R. Loui. "A Mathematical Treatment of Defeasible Reasoning and its Implementation," *Artificial Intelligence 52*, 1992.

Stein, L. "Resolving ambiguity in nonmonotonic reasoning," dissertation, Brown University, 1990.

Stillman, J. "It's not my default: the complexity of membership problems in restricted propositional default logic," *Proc. AAAI*, 1990.

Touretzky, D., R. Thomason, and J. Horty. "A skeptic's menagerie: conflictors, preemptors, reinstaters, and zombies in nonmonotonic inheritance," *Proc. IJCAI*, 1991.

Vreeswijk, G. "Feasibility of defeat in defeasible reasoning," *Proc. KR*, 1991.

```
INPUT:
on(a,b,0)!
on(b,c,0)!
on(c,table,0)!
clear(a,0)!
A s clear(table,s).
R x,y,z,s on(x,z,s) ^ clear(x,s) ^ clear(y,s) >- on(x,y,move(x,y,s)).
R x,y,z,s clear(x,s) ^ clear(y,s) ^ on(x,z,s) >- clear(z,move(x,y,s)).
R x,y,s on(x,y,s) >- ~clear(y,s).
R x,y,z,t,s on(x,y,s) >- on(x,y,move(z,t,s)).
R x,z,t,s clear(x,s) >- clear(x,move(z,t,s)).
clear(b,move(a,table,0))?
DISPUTING clear(b,move(a,table,0))

A tries to support:  clear(b,move(a,table,0))

A has an argument for clear(b,move(a,table,0)):   on(a,b,0) clear(a,0)
B has an argument for ~(clear(b,move(a,table,0))):   on(a,b,0)
   B is CHECKING SUBARGUMENTS to block A's establishment of clear(b,move(a,table,0))
   TARGETS:  ~(clear(b,move(a,table,0)))
   NEW TARGET for B:  ~(clear(b,move(a,table,0)))
   ~(clear(b,move(a,table,0))) already PENDING: B backtracks
   B ALLOWS A's subarguments for clear(b,move(a,table,0))
      A is CHECKING SUBARGUMENTS to defeat B's establishment of ~(clear(b,move(a,table,0
)))
      TARGETS:  ~(on(a,b,move(a,table,0)))    clear(b,move(a,table,0))
      NEW TARGET for A:  ~(on(a,b,move(a,table,0)))
      A is ESTABLISHING ~(on(a,b,move(a,table,0))) for support
      A has no argument for ~(on(a,b,move(a,table,0)))
      A FAILED TO ESTABLISH ~(on(a,b,move(a,table,0))) for support
      NEW TARGET for A:  clear(b,move(a,table,0))
      clear(b,move(a,table,0)) already PENDING: A backtracks
      A ALLOWS B's subarguments for ~(clear(b,move(a,table,0)))

************ round 1 ************
A's ARGUMENT:  (1)B's ARGUMENT:  (-1)
A ACTIVATOR: (clear(a,0) ^ (clear(table,0) ^ on(a,b,0)))
B ACTIVATOR: on(a,b,move(a,table,0))
A's ARGUMENT IS MORE SPECIFIC
no more arguments for B
A WINS clear(b,move(a,table,0))
1
4.883138 seconds
```

```
INPUT:
bird(tweety)!
~flies(tweety)!
R x bird(x) >- flies(x).
flies(tweety)?
DISPUTING flies(tweety)

A tries to support:  flies(tweety)

A has an argument for flies(tweety):   bird(tweety)
B has an argument for ~(flies(tweety)):   ~flies(tweety)
   B is CHECKING SUBARGUMENTS to block A's establishment of flies(tweety)
   TARGETS:  ~(flies(tweety))
   NEW TARGET for B:  ~(flies(tweety))
   ~(flies(tweety)) already PENDING: B backtracks
   B ALLOWS A's subarguments for flies(tweety)
      A is CHECKING SUBARGUMENTS to defeat B's establishment of ~(flies(tweety))
      TARGETS:  no targets
      A ALLOWS B's subarguments for ~(flies(tweety))

************ round 1 ************
A's ARGUMENT:  (1)B's ARGUMENT:  (-1)
A ACTIVATOR: bird(tweety)
B ACTIVATOR: (NULL)
B's ARGUMENT IS MORE SPECIFIC
no more arguments for A
A exhausts arguments for: flies(tweety)

B tries to support: ~(flies(tweety))

   A is CHECKING SUBARGUMENTS to block B's establishment of ~(flies(tweety))
   TARGETS:  no targets
   A ALLOWS B's subarguments for ~(flies(tweety))
      B is CHECKING SUBARGUMENTS to defeat A's establishment of flies(tweety)
      TARGETS:  ~(flies(tweety))
      NEW TARGET for B:  ~(flies(tweety))
      ~(flies(tweety)) already PENDING: B backtracks
      B ALLOWS A's subarguments for flies(tweety)

************ round 2 ************
B's ARGUMENT:  (1)A's ARGUMENT:  (-1)already argued; B won
no more arguments for A
B WINS ~(flies(tweety))
-1
0.049998 seconds
```

```
INPUT:
penguin(opus)!
R x bird(x) >- flies(x).
R x penguin(x) >- ~flies(x).
A x (penguin(x) => bird(x)).
flies(opus)?
DISPUTING flies(opus)

A tries to support:  flies(opus)

A has an argument for flies(opus):   penguin(opus)
B has an argument for ~(flies(opus)):   penguin(opus)
   B is CHECKING SUBARGUMENTS to block A's establishment of flies(opus)
   TARGETS:  ~(flies(opus))
   NEW TARGET for B:  ~(flies(opus))
   ~(flies(opus)) already PENDING: B backtracks
   B ALLOWS A's subarguments for flies(opus)
      A is CHECKING SUBARGUMENTS to defeat B's establishment of ~(flies(opus))
      TARGETS:  flies(opus)
      NEW TARGET for A:  flies(opus)
      flies(opus) already PENDING: A backtracks
      A ALLOWS B's subarguments for ~(flies(opus))

************ round 1 ************
A's ARGUMENT:  (1)B's ARGUMENT:  (-1)
A ACTIVATOR: bird(opus)
B ACTIVATOR: penguin(opus)
B's ARGUMENT IS MORE SPECIFIC
no more arguments for A
A exhausts arguments for: flies(opus)

B tries to support: ~(flies(opus))

   A is CHECKING SUBARGUMENTS to block B's establishment of ~(flies(opus))
   TARGETS:  flies(opus)
   NEW TARGET for A:  flies(opus)
   flies(opus) already PENDING: A backtracks
   A ALLOWS B's subarguments for ~(flies(opus))
      B is CHECKING SUBARGUMENTS to defeat A's establishment of flies(opus)
      TARGETS:  ~(flies(opus))
      NEW TARGET for B:  ~(flies(opus))
      ~(flies(opus)) already PENDING: B backtracks
      B ALLOWS A's subarguments for flies(opus)

************ round 2 ************
B's ARGUMENT:  (1)A's ARGUMENT:  (-1)already argued; B won
no more arguments for A
B WINS ~(flies(opus))
-1
0.099996 seconds
```

```
INPUT:
cat(garfield)!
R x cat(x) >- aloof(x).
R x aloof(x) >- ~likes-people(x).
R x cat(x) >- likes-people(x).
likes-people(garfield)?
DISPUTING likes-people(garfield)

A tries to support:  likes-people(garfield)

A has an argument for likes-people(garfield):   cat(garfield)
B has an argument for ~(likes-people(garfield)):   cat(garfield)
   B is CHECKING SUBARGUMENTS to block A's establishment of likes-people(garfield)
   TARGETS:  ~(likes-people(garfield))
   NEW TARGET for B:  ~(likes-people(garfield))
   ~(likes-people(garfield)) already PENDING: B backtracks
   B ALLOWS A's subarguments for likes-people(garfield)
      A is CHECKING SUBARGUMENTS to defeat B's establishment of ~(likes-people(garfield)
)
      TARGETS:  ~(aloof(garfield))    likes-people(garfield)
      NEW TARGET for A:  ~(aloof(garfield))
      A is ESTABLISHING ~(aloof(garfield)) for support
      A has no argument for ~(aloof(garfield))
      A FAILED TO ESTABLISH ~(aloof(garfield)) for support
      NEW TARGET for A:  likes-people(garfield)
      likes-people(garfield) already PENDING: A backtracks
      A ALLOWS B's subarguments for ~(likes-people(garfield))

************ round 1 ************
A's ARGUMENT:  (1)B's ARGUMENT:  (-1)
A ACTIVATOR: cat(garfield)
B ACTIVATOR: aloof(garfield)
A's ARGUMENT IS MORE SPECIFIC
no more arguments for B
A WINS likes-people(garfield)
1
0.083330 seconds
```

```
INPUT:
republican(nixon) ^ quaker(nixon)!
R x republican(x) >- hawk(x).
R x quaker(x) >- dove(x).
A x (hawk(x) <=> ~dove(x)).
dove(nixon)?
DISPUTING dove(nixon)

A tries to support:  dove(nixon)

A has an argument for dove(nixon):   quaker(nixon)
B has an argument for ~(dove(nixon)):   republican(nixon)
   B is CHECKING SUBARGUMENTS to block A's establishment of dove(nixon)
   TARGETS:  ~(dove(nixon))
   NEW TARGET for B:  ~(dove(nixon))
   ~(dove(nixon)) already PENDING: B backtracks
   B ALLOWS A's subarguments for dove(nixon)
      A is CHECKING SUBARGUMENTS to defeat B's establishment of ~(dove(nixon))
      TARGETS:  ~(hawk(nixon))
      NEW TARGET for A:  ~(hawk(nixon))
      A is ESTABLISHING ~(hawk(nixon)) for support
      A has an argument for ~(hawk(nixon)):         quaker(nixon)
         B is CHECKING SUBARGUMENTS to block A's establishment of ~(hawk(nixon))
         TARGETS:  ~(dove(nixon))
         NEW TARGET for B:  ~(dove(nixon))
         B is ESTABLISHING ~(dove(nixon)) for viability
         B has an argument for ~(dove(nixon)):            republican(nixon)
            A is CHECKING SUBARGUMENTS to defeat B's establishment of ~(dove(nixon))
            TARGETS:  ~(hawk(nixon))
            NEW TARGET for A:  ~(hawk(nixon))
            ~(hawk(nixon)) already PENDING: A backtracks
            A ALLOWS B's subarguments for ~(dove(nixon))
            A CHECKING FOR TOP-LEVEL COUNTERARGS to ~(dove(nixon))
            A has an argument for dove(nixon):              quaker(nixon)
               B is CHECKING SUBARGUMENTS to defeat A's establishment of dove(nixon)
               TARGETS:  ~(dove(nixon))
               NEW TARGET for B:  ~(dove(nixon))
               ~(dove(nixon)) already PENDING: B backtracks
               B ALLOWS A's subarguments for dove(nixon)

************ round 1 ************
         B's ARGUMENT:  (1)         A's ARGUMENT:  (-1)
         B ACTIVATOR: republican(nixon)
         A ACTIVATOR: quaker(nixon)
         INCONCLUSIVE
         A seeks another argument against ~(dove(nixon))
         no more arguments for A
         B ESTABLISHED ~(dove(nixon)) for viability
         B DISALLOWS A's subarguments for ~(hawk(nixon))
      A seeks another argument for ~(hawk(nixon))
      no more arguments for A
      A FAILED TO ESTABLISH ~(hawk(nixon)) for support
      A ALLOWS B's subarguments for ~(dove(nixon))

************ round 1 ************
A's ARGUMENT:  (1)B's ARGUMENT:  (-1)
A ACTIVATOR: quaker(nixon)
B ACTIVATOR: republican(nixon)
INCONCLUSIVE
no more arguments for A
A exhausts arguments for: dove(nixon)

B tries to support: ~(dove(nixon))

   A is CHECKING SUBARGUMENTS to block B's establishment of ~(dove(nixon))
   TARGETS:  ~(hawk(nixon))
   NEW TARGET for A:  ~(hawk(nixon))
   A is ESTABLISHING ~(hawk(nixon)) for viability
   A has an argument for ~(hawk(nixon)):      quaker(nixon)
      B is CHECKING SUBARGUMENTS to defeat A's establishment of ~(hawk(nixon))
      TARGETS:  ~(dove(nixon))
      NEW TARGET for B:  ~(dove(nixon))
      ~(dove(nixon)) already PENDING: B backtracks
      B ALLOWS A's subarguments for ~(hawk(nixon))
      B CHECKING FOR TOP-LEVEL COUNTERARGS to ~(hawk(nixon))
      B has an argument for hawk(nixon):         republican(nixon)
         A is CHECKING SUBARGUMENTS to defeat B's establishment of hawk(nixon)
         TARGETS:  ~(hawk(nixon))
         NEW TARGET for A:  ~(hawk(nixon))
         ~(hawk(nixon)) already PENDING: A backtracks
         A ALLOWS B's subarguments for hawk(nixon)

************ round 1 ************
   A's ARGUMENT:  (1)   B's ARGUMENT:  (-1)
   A ACTIVATOR: quaker(nixon)
   B ACTIVATOR: republican(nixon)
   INCONCLUSIVE
   B seeks another argument against ~(hawk(nixon))
   no more arguments for B
   A ESTABLISHED ~(hawk(nixon)) for viability
   A DISALLOWS B's subarguments for ~(dove(nixon))
   A is CHECKING SUBARGUMENTS to block B's establishment of ~(dove(nixon))
   TARGETS:  ~(hawk(nixon))
   NEW TARGET for A:  ~(hawk(nixon))
   ~(hawk(nixon)) already PENDING: A backtracks
   A ALLOWS B's subarguments for ~(dove(nixon))
already countered  (argument -1)
no more arguments for B
NO WINNER FOR dove(nixon)
0
0.233324 seconds
```

```
INPUT:
republican(nixon) ^ quaker(nixon)!
R x republican(x) >- hawk(x).
R x quaker(x) >- dove(x).
A x (hawk(x) <=> ~dove(x)).
A x (hawk(x) v dove(x) <=> politically-motivated(x)).
politically-motivated(nixon)?
DISPUTING politically-motivated(nixon)

A tries to support:  politically-motivated(nixon)

A has an argument for politically-motivated(nixon):   case 1  case 2
B has no argument for ~(politically-motivated(nixon))
   B is CHECKING SUBARGUMENTS to block A's establishment of politically-motivated(nixon)
   TARGETS:  no targets
   B ALLOWS A's subarguments for politically-motivated(nixon)
A WINS politically-motivated(nixon)
1
3.766516 seconds
```

```
INPUT:
republican(nixon) ^ quaker(nixon)!
R x republican(x) >- ~pacifist(x).
R x quaker(x) >- pacifist(x).
R x republican(x) >- footballfan(x).
R x footballfan(x) >- ~antimilitary(x).
R x pacifist(x) >- antimilitary(x).
antimilitary(nixon)?
DISPUTING antimilitary(nixon)

A tries to support:  antimilitary(nixon)

A has an argument for antimilitary(nixon):   quaker(nixon)
B has an argument for ~(antimilitary(nixon)):   republican(nixon)
   B is CHECKING SUBARGUMENTS to block A's establishment of antimilitary(nixon)
   TARGETS:  ~(pacifist(nixon))    ~(antimilitary(nixon))
   NEW TARGET for B:  ~(pacifist(nixon))
   B is ESTABLISHING ~(pacifist(nixon)) for viability
   B has an argument for ~(pacifist(nixon)):      republican(nixon)
      A is CHECKING SUBARGUMENTS to defeat B's establishment of ~(pacifist(nixon))
      TARGETS:  pacifist(nixon)
      NEW TARGET for A:  pacifist(nixon)
      pacifist(nixon) already PENDING: A backtracks
      A ALLOWS B's subarguments for ~(pacifist(nixon))
      A CHECKING FOR TOP-LEVEL COUNTERARGS to ~(pacifist(nixon))
      A has an argument for pacifist(nixon):         quaker(nixon)
         B is CHECKING SUBARGUMENTS to defeat A's establishment of pacifist(nixon)
         TARGETS:  ~(pacifist(nixon))
         NEW TARGET for B:  ~(pacifist(nixon))
         ~(pacifist(nixon)) already PENDING: B backtracks
         B ALLOWS A's subarguments for pacifist(nixon)

************ round 1 ************
   B's ARGUMENT:  (1)   A's ARGUMENT:  (-1)
   B ACTIVATOR: republican(nixon)
   A ACTIVATOR: quaker(nixon)
   INCONCLUSIVE
   A seeks another argument against ~(pacifist(nixon))
   no more arguments for A
   B ESTABLISHED ~(pacifist(nixon)) for viability
   B DISALLOWS A's subarguments for antimilitary(nixon)
no more arguments for A
A exhausts arguments for: antimilitary(nixon)

B tries to support: ~(antimilitary(nixon))

   A is CHECKING SUBARGUMENTS to block B's establishment of ~(antimilitary(nixon))
   TARGETS:  ~(footballfan(nixon))    antimilitary(nixon)
   NEW TARGET for A:  ~(footballfan(nixon))
   A is ESTABLISHING ~(footballfan(nixon)) for viability
   A has no argument for ~(footballfan(nixon))
   A FAILED TO ESTABLISH ~(footballfan(nixon)) for viability
   NEW TARGET for A:  antimilitary(nixon)
   antimilitary(nixon) already PENDING: A backtracks
   A ALLOWS B's subarguments for ~(antimilitary(nixon))
      B is CHECKING SUBARGUMENTS to defeat A's establishment of antimilitary(nixon)
      TARGETS:  ~(pacifist(nixon))    ~(antimilitary(nixon))
      NEW TARGET for B:  ~(pacifist(nixon))
      B is ESTABLISHING ~(pacifist(nixon)) for support
      B has an argument for ~(pacifist(nixon)):         republican(nixon)
         A is CHECKING SUBARGUMENTS to block B's establishment of ~(pacifist(nixon))
         TARGETS:  pacifist(nixon)
         NEW TARGET for A:  pacifist(nixon)
         pacifist(nixon) already PENDING: A backtracks
         A ALLOWS B's subarguments for ~(pacifist(nixon))
         A CHECKING FOR TOP-LEVEL COUNTERARGS to ~(pacifist(nixon))
         A has an argument for pacifist(nixon):           quaker(nixon)
            B is CHECKING SUBARGUMENTS to defeat A's establishment of pacifist(nixon)
            TARGETS:  ~(pacifist(nixon))
            NEW TARGET for B:  ~(pacifist(nixon))
            ~(pacifist(nixon)) already PENDING: B backtracks
            B ALLOWS A's subarguments for pacifist(nixon)

************ round 1 ************
      B's ARGUMENT:  (1)      A's ARGUMENT:  (-1)
      B ACTIVATOR: republican(nixon)
      A ACTIVATOR: quaker(nixon)
      INCONCLUSIVE
      B seeks another argument for ~(pacifist(nixon))
      no more arguments for B
      B FAILED TO ESTABLISH ~(pacifist(nixon)) for support
      NEW TARGET for B:  ~(antimilitary(nixon))
      ~(antimilitary(nixon)) already PENDING: B backtracks
      B ALLOWS A's subarguments for antimilitary(nixon)

************ round 1 ************
B's ARGUMENT:  (1)A's ARGUMENT:  (-1)
B ACTIVATOR: pacifist(nixon)
A ACTIVATOR: footballfan(nixon)
INCONCLUSIVE
no more arguments for B
B exhausts arguments for: ~(antimilitary(nixon))
NO WINNER FOR antimilitary(nixon)
0
0.316654 seconds
```

```
INPUT:
alive(0) ^ loaded(s(0)) ^ fired(s(s(s(0))))!
R x alive(x) >- alive(s(x)).
R x loaded(x) >- loaded(s(x)).
R x loaded(x) ^ fired(x) ^ alive(x) >- ~alive(s(x)).
~alive(s(s(s(s(0)))))?
DISPUTING ~(alive(s(s(s(s(0))))))

A tries to support:  ~(alive(s(s(s(s(0))))))

A has an argument for ~(alive(s(s(s(s(0)))))):   alive(0) loaded(s(0)) fired(s(s(s(0))))
B has an argument for alive(s(s(s(s(0))))):   alive(0)
   B is CHECKING SUBARGUMENTS to block A's establishment of ~(alive(s(s(s(s(0))))))
   TARGETS:  ~(alive(s(0)))    ~(alive(s(s(0))))    ~(alive(s(s(s(0)))))    ~(loaded(s(s
(0))))    ~(loaded(s(s(s(0)))))    alive(s(s(s(s(0)))))
   NEW TARGET for B:  ~(alive(s(0)))
   B is ESTABLISHING ~(alive(s(0))) for viability
   B has no argument for ~(alive(s(0)))
   B FAILED TO ESTABLISH ~(alive(s(0))) for viability
   NEW TARGET for B:  ~(alive(s(s(0))))
   B is ESTABLISHING ~(alive(s(s(0)))) for viability
   B has no argument for ~(alive(s(s(0))))
   B FAILED TO ESTABLISH ~(alive(s(s(0)))) for viability
   NEW TARGET for B:  ~(alive(s(s(s(0)))))
   B is ESTABLISHING ~(alive(s(s(s(0))))) for viability
   B has no argument for ~(alive(s(s(s(0)))))
   B FAILED TO ESTABLISH ~(alive(s(s(s(0))))) for viability
   NEW TARGET for B:  ~(loaded(s(s(0))))
   B is ESTABLISHING ~(loaded(s(s(0)))) for viability
   B has no argument for ~(loaded(s(s(0))))
   B FAILED TO ESTABLISH ~(loaded(s(s(0)))) for viability
   NEW TARGET for B:  ~(loaded(s(s(s(0)))))
   B is ESTABLISHING ~(loaded(s(s(s(0))))) for viability
   B has no argument for ~(loaded(s(s(s(0)))))
   B FAILED TO ESTABLISH ~(loaded(s(s(s(0))))) for viability
   NEW TARGET for B:  alive(s(s(s(s(0)))))
   alive(s(s(s(s(0))))) already PENDING: B backtracks
   B ALLOWS A's subarguments for ~(alive(s(s(s(s(0))))))
      A is CHECKING SUBARGUMENTS to defeat B's establishment of alive(s(s(s(s(0)))))
      TARGETS:  ~(alive(s(0)))    ~(alive(s(s(0))))    ~(alive(s(s(s(0)))))    ~(alive(s
(s(s(s(0))))))
      NEW TARGET for A:  ~(alive(s(0)))
      A is ESTABLISHING ~(alive(s(0))) for support
      A has no argument for ~(alive(s(0)))
      A FAILED TO ESTABLISH ~(alive(s(0))) for support
      NEW TARGET for A:  ~(alive(s(s(0))))
      A is ESTABLISHING ~(alive(s(s(0)))) for support
      A has no argument for ~(alive(s(s(0))))
      A FAILED TO ESTABLISH ~(alive(s(s(0)))) for support
      NEW TARGET for A:  ~(alive(s(s(s(0)))))
      A is ESTABLISHING ~(alive(s(s(s(0))))) for support
      A has no argument for ~(alive(s(s(s(0)))))
      A FAILED TO ESTABLISH ~(alive(s(s(s(0))))) for support
      NEW TARGET for A:  ~(alive(s(s(s(s(0))))))
      ~(alive(s(s(s(s(0)))))) already PENDING: A backtracks
      A ALLOWS B's subarguments for alive(s(s(s(s(0)))))

************ round 1 ************
A's ARGUMENT:  (1)B's ARGUMENT:  (-1)
A ACTIVATOR: (loaded(s(s(s(0)))) ^ (fired(s(s(s(0)))) ^ alive(s(s(s(0))))))
B ACTIVATOR: alive(s(s(s(0))))
A's ARGUMENT IS MORE SPECIFIC
no more arguments for B
A WINS ~(alive(s(s(s(s(0))))))
1
3.666520 seconds
```

```
INPUT:
alive(0) ^ loaded(s(0)) ^ fired(s(s(s(0)))))!
R x alive(x) >- alive(s(x)).
R x loaded(x) >- loaded(s(x)).
R x loaded(x) ^ fired(x) ^ alive(x) >- ~alive(s(x)).
E x (~alive(x))?
DISPUTING Exists x (~(alive(x)))

A tries to support:  Exists x (~(alive(x)))

A has an argument for Exists x (~(alive(x))):   alive(0) loaded(s(0)) fired(s(s(s(0))))
B has no argument for All x (alive(x))
   B is CHECKING SUBARGUMENTS to block A's establishment of Exists x (~(alive(x)))
   TARGETS:  ~(alive(s(0)))    ~(alive(s(s(0))))    ~(alive(s(s(s(0)))))    ~(loaded(s(s
(0))))    ~(loaded(s(s(s(0)))))    alive(s(s(s(s(0)))))
   NEW TARGET for B:  ~(alive(s(0)))
   B is ESTABLISHING ~(alive(s(0))) for viability
   B has no argument for ~(alive(s(0)))
   B FAILED TO ESTABLISH ~(alive(s(0))) for viability
   NEW TARGET for B:  ~(alive(s(s(0))))
   B is ESTABLISHING ~(alive(s(s(0)))) for viability
   B has no argument for ~(alive(s(s(0))))
   B FAILED TO ESTABLISH ~(alive(s(s(0)))) for viability
   NEW TARGET for B:  ~(alive(s(s(s(0)))))
   B is ESTABLISHING ~(alive(s(s(s(0))))) for viability
   B has no argument for ~(alive(s(s(s(0)))))
   B FAILED TO ESTABLISH ~(alive(s(s(s(0))))) for viability
   NEW TARGET for B:  ~(loaded(s(s(0))))
   B is ESTABLISHING ~(loaded(s(s(0)))) for viability
   B has no argument for ~(loaded(s(s(0))))
   B FAILED TO ESTABLISH ~(loaded(s(s(0)))) for viability
   NEW TARGET for B:  ~(loaded(s(s(s(0)))))
   B is ESTABLISHING ~(loaded(s(s(s(0))))) for viability
   B has no argument for ~(loaded(s(s(s(0)))))
   B FAILED TO ESTABLISH ~(loaded(s(s(s(0))))) for viability
   NEW TARGET for B:  alive(s(s(s(s(0)))))
   B is ESTABLISHING alive(s(s(s(s(0))))) for viability
   B has an argument for alive(s(s(s(s(0))))):      alive(0)
      A is CHECKING SUBARGUMENTS to defeat B's establishment of alive(s(s(s(s(0)))))
      TARGETS:  ~(alive(s(0)))    ~(alive(s(s(0))))    ~(alive(s(s(s(0)))))    ~(alive(s
(s(s(s(0))))))
      NEW TARGET for A:  ~(alive(s(0)))
      A is ESTABLISHING ~(alive(s(0))) for support
      A has no argument for ~(alive(s(0)))
      A FAILED TO ESTABLISH ~(alive(s(0))) for support
      NEW TARGET for A:  ~(alive(s(s(0))))
      A is ESTABLISHING ~(alive(s(s(0)))) for support
      A has no argument for ~(alive(s(s(0))))
      A FAILED TO ESTABLISH ~(alive(s(s(0)))) for support
      NEW TARGET for A:  ~(alive(s(s(s(0)))))
      A is ESTABLISHING ~(alive(s(s(s(0))))) for support
      A has no argument for ~(alive(s(s(s(0)))))
      A FAILED TO ESTABLISH ~(alive(s(s(s(0))))) for support
      NEW TARGET for A:  ~(alive(s(s(s(s(0))))))
      ~(alive(s(s(s(s(0))))))) already PENDING: A backtracks
      A ALLOWS B's subarguments for alive(s(s(s(s(0)))))
      A CHECKING FOR TOP-LEVEL COUNTERARGS to alive(s(s(s(s(0)))))
      A has an argument for ~(alive(s(s(s(s(0)))))):        alive(0)       loaded(s(0))
       fired(s(s(s(0))))
        B is CHECKING SUBARGUMENTS to defeat A's establishment of ~(alive(s(s(s(s(0))))
))
        TARGETS:  ~(alive(s(0)))    ~(alive(s(s(0))))    ~(alive(s(s(s(0)))))    ~(load
ed(s(s(0))))    ~(loaded(s(s(s(0)))))    alive(s(s(s(s(0)))))
        NEW TARGET for B:  ~(alive(s(0)))
        B is ESTABLISHING ~(alive(s(0))) for support
        B has no argument for ~(alive(s(0)))
        B FAILED TO ESTABLISH ~(alive(s(0))) for support
        NEW TARGET for B:  ~(alive(s(s(0))))
        B is ESTABLISHING ~(alive(s(s(0)))) for support
        B has no argument for ~(alive(s(s(0))))
        B FAILED TO ESTABLISH ~(alive(s(s(0)))) for support
        NEW TARGET for B:  ~(alive(s(s(s(0)))))
        B is ESTABLISHING ~(alive(s(s(s(0))))) for support
        B has no argument for ~(alive(s(s(s(0)))))
        B FAILED TO ESTABLISH ~(alive(s(s(s(0))))) for support
        NEW TARGET for B:  ~(alive(s(s(s(s(0))))))
        B is ESTABLISHING ~(alive(s(s(s(s(0)))))) for support
        B has no argument for ~(alive(s(s(s(s(0))))))
        B FAILED TO ESTABLISH ~(alive(s(s(s(s(0)))))) for support
        NEW TARGET for B:  ~(loaded(s(s(0))))
        B is ESTABLISHING ~(loaded(s(s(0)))) for support
        B has no argument for ~(loaded(s(s(0))))
        B FAILED TO ESTABLISH ~(loaded(s(s(0)))) for support
        NEW TARGET for B:  ~(loaded(s(s(s(0)))))
        B is ESTABLISHING ~(loaded(s(s(s(0))))) for support
        B has no argument for ~(loaded(s(s(s(0)))))
        B FAILED TO ESTABLISH ~(loaded(s(s(s(0))))) for support
        NEW TARGET for B:  alive(s(s(s(s(0)))))
        alive(s(s(s(s(0))))) already PENDING: B backtracks
        B ALLOWS A's subarguments for ~(alive(s(s(s(s(0))))))

************ round 1 ************
   B's ARGUMENT:  (1)   A's ARGUMENT:  (-1)
   B ACTIVATOR: alive(s(s(s(0))))
   A ACTIVATOR: (loaded(s(s(s(0)))) ^ (fired(s(s(s(0)))) ^ alive(s(s(s(0))))))
   A's ARGUMENT IS MORE SPECIFIC
   B seeks another argument for alive(s(s(s(s(0)))))
   no more arguments for B
   B FAILED TO ESTABLISH alive(s(s(s(s(0))))) for viability
   B ALLOWS A's subarguments for Exists x (~(alive(x)))
A WINS Exists x (~(alive(x)))
1
7.066384 seconds
```

```
INPUT:
alive(0) ^ loaded(s(0)) ^ fired(s(s(s(0)))))!
R x alive(x) >- alive(s(x)).
R x loaded(x) >- loaded(s(x)).
R x loaded(x) ^ fired(x) ^ alive(x) >- ~alive(s(x)).
R x alive(x) >- saysyabbadabbadoo(x).
saysyabbadabbadoo(s(s(s(s(0))))))?
DISPUTING saysyabbadabbadoo(s(s(s(s(0)))))

A tries to support:  saysyabbadabbadoo(s(s(s(s(0)))))

A has an argument for saysyabbadabbadoo(s(s(s(s(0))))):   alive(0)
B has no argument for ~(saysyabbadabbadoo(s(s(s(s(0))))))
   B is CHECKING SUBARGUMENTS to block A's establishment of saysyabbadabbadoo(s(s(s(s(0
))))
   TARGETS:  ~(alive(s(0)))    ~(alive(s(s(0))))    ~(alive(s(s(s(0)))))    ~(alive(s(s(
s(s(0))))))    ~(saysyabbadabbadoo(s(s(s(s(0))))))
   NEW TARGET for B:  ~(alive(s(0)))
   B is ESTABLISHING ~(alive(s(0))) for viability
   B has no argument for ~(alive(s(0)))
   B FAILED TO ESTABLISH ~(alive(s(0))) for viability
   NEW TARGET for B:  ~(alive(s(s(0))))
   B is ESTABLISHING ~(alive(s(s(0)))) for viability
   B has no argument for ~(alive(s(s(0))))
   B FAILED TO ESTABLISH ~(alive(s(s(0)))) for viability
   NEW TARGET for B:  ~(alive(s(s(s(0)))))
   B is ESTABLISHING ~(alive(s(s(s(0))))) for viability
   B has no argument for ~(alive(s(s(s(0)))))
   B FAILED TO ESTABLISH ~(alive(s(s(s(0))))) for viability
   NEW TARGET for B:  ~(alive(s(s(s(s(0))))))
   B is ESTABLISHING ~(alive(s(s(s(s(0)))))) for viability
   B has an argument for ~(alive(s(s(s(s(0)))))):    alive(0)    loaded(s(0))    fired
(s(s(s(0))))
      A is CHECKING SUBARGUMENTS to defeat B's establishment of ~(alive(s(s(s(s(0))))))
      TARGETS:  ~(alive(s(0)))    ~(alive(s(s(0))))    ~(alive(s(s(s(0)))))    ~(loaded(
s(s(0))))    ~(loaded(s(s(s(0)))))    alive(s(s(s(s(0)))))
      NEW TARGET for A:  ~(alive(s(0)))
      A is ESTABLISHING ~(alive(s(0))) for support
      A has no argument for ~(alive(s(0)))
      A FAILED TO ESTABLISH ~(alive(s(0))) for support
      NEW TARGET for A:  ~(alive(s(s(0))))
      A is ESTABLISHING ~(alive(s(s(0)))) for support
      A has no argument for ~(alive(s(s(0))))
      A FAILED TO ESTABLISH ~(alive(s(s(0)))) for support
      NEW TARGET for A:  ~(alive(s(s(s(0)))))
      A is ESTABLISHING ~(alive(s(s(s(0))))) for support
      A has no argument for ~(alive(s(s(s(0)))))
      A FAILED TO ESTABLISH ~(alive(s(s(s(0))))) for support
      NEW TARGET for A:  ~(loaded(s(s(0))))
      A is ESTABLISHING ~(loaded(s(s(0)))) for support
      A has no argument for ~(loaded(s(s(0))))
      A FAILED TO ESTABLISH ~(loaded(s(s(0)))) for support
      NEW TARGET for A:  ~(loaded(s(s(s(0)))))
      A is ESTABLISHING ~(loaded(s(s(s(0))))) for support
      A has no argument for ~(loaded(s(s(s(0)))))
      A FAILED TO ESTABLISH ~(loaded(s(s(s(0))))) for support
      NEW TARGET for A:  alive(s(s(s(s(0)))))
      alive(s(s(s(s(0))))) already PENDING: A backtracks
      A ALLOWS B's subarguments for ~(alive(s(s(s(s(0))))))
      A CHECKING FOR TOP-LEVEL COUNTERARGS to ~(alive(s(s(s(s(0))))))
      A has an argument for alive(s(s(s(s(0))))):    alive(0)
         B is CHECKING SUBARGUMENTS to defeat A's establishment of alive(s(s(s(s(0)))))
         TARGETS:  ~(alive(s(0)))    ~(alive(s(s(0))))    ~(alive(s(s(s(0)))))    ~(aliv
e(s(s(s(s(0))))))
         NEW TARGET for B:  ~(alive(s(0)))
         B is ESTABLISHING ~(alive(s(0))) for support
         B has no argument for ~(alive(s(0)))
         B FAILED TO ESTABLISH ~(alive(s(0))) for support
         NEW TARGET for B:  ~(alive(s(s(0))))
         B is ESTABLISHING ~(alive(s(s(0)))) for support
         B has no argument for ~(alive(s(s(0))))
         B FAILED TO ESTABLISH ~(alive(s(s(0)))) for support
         NEW TARGET for B:  ~(alive(s(s(s(0)))))
         B is ESTABLISHING ~(alive(s(s(s(0))))) for support
         B has no argument for ~(alive(s(s(s(0)))))
         B FAILED TO ESTABLISH ~(alive(s(s(s(0))))) for support
         NEW TARGET for B:  ~(alive(s(s(s(s(0))))))
         ~(alive(s(s(s(s(0)))))) already PENDING: B backtracks
         B ALLOWS A's subarguments for alive(s(s(s(s(0)))))

************ round 1 ************
   B's ARGUMENT:  (1)   A's ARGUMENT:  (-1)
   B ACTIVATOR: (loaded(s(s(s(0)))) ^ (fired(s(s(s(0)))) ^ alive(s(s(s(0))))))
   A ACTIVATOR: alive(s(s(s(0))))
   B's ARGUMENT IS MORE SPECIFIC
   A seeks another argument against ~(alive(s(s(s(s(0))))))
   no more arguments for A
   B ESTABLISHED ~(alive(s(s(s(s(0)))))) for viability
   B DISALLOWS A's subarguments for saysyabbadabbadoo(s(s(s(s(0)))))
no more arguments for A
A exhausts arguments for: saysyabbadabbadoo(s(s(s(s(0)))))

B tries to support: ~(saysyabbadabbadoo(s(s(s(s(0))))))

NO WINNER FOR saysyabbadabbadoo(s(s(s(s(0)))))
0
4.216498 seconds
```

```
INPUT:
dancer(noemi)!
R x dancer(x) >- graceful(x).
R x dancer(x) >- ~ballerina(x).
R x graceful(x) ^ dancer(x) >- ballerina(x).
ballerina(noemi)?
DISPUTING ballerina(noemi)

A tries to support:  ballerina(noemi)

A has an argument for ballerina(noemi):   dancer(noemi)
B has an argument for ~(ballerina(noemi)):   dancer(noemi)
   B is CHECKING SUBARGUMENTS to block A's establishment of ballerina(noemi)
   TARGETS:  ~(graceful(noemi))    ~(ballerina(noemi))
   NEW TARGET for B:  ~(graceful(noemi))
   B is ESTABLISHING ~(graceful(noemi)) for viability
   B has no argument for ~(graceful(noemi))
   B FAILED TO ESTABLISH ~(graceful(noemi)) for viability
   NEW TARGET for B:  ~(ballerina(noemi))
   ~(ballerina(noemi)) already PENDING: B backtracks
   B ALLOWS A's subarguments for ballerina(noemi)
      A is CHECKING SUBARGUMENTS to defeat B's establishment of ~(ballerina(noemi))
      TARGETS:  ballerina(noemi)
      NEW TARGET for A:  ballerina(noemi)
      ballerina(noemi) already PENDING: A backtracks
      A ALLOWS B's subarguments for ~(ballerina(noemi))

************ round 1 ************
A's ARGUMENT:  (1)B's ARGUMENT:  (-1)
A ACTIVATOR: (graceful(noemi) ^ dancer(noemi))
B ACTIVATOR: dancer(noemi)
INCONCLUSIVE
no more arguments for A
A exhausts arguments for: ballerina(noemi)

B tries to support: ~(ballerina(noemi))

   A is CHECKING SUBARGUMENTS to block B's establishment of ~(ballerina(noemi))
   TARGETS:  ballerina(noemi)
   NEW TARGET for A:  ballerina(noemi)
   ballerina(noemi) already PENDING: A backtracks
   A ALLOWS B's subarguments for ~(ballerina(noemi))
already countered  (argument -1)
no more arguments for B
NO WINNER FOR ballerina(noemi)
0
0.116662 seconds
```

```
INPUT:
republican(nixon) ^ quaker(nixon)!
R x republican(x) ^ quaker(x) >- ~pacifist(x).
R x quaker(x) >- pacifist(x).
R x republican(x) >- footballfan(x).
R x footballfan(x) >- ~antimilitary(x).
R x pacifist(x) >- antimilitary(x).
~antimilitary(nixon)?
DISPUTING ~(antimilitary(nixon))

A tries to support: ~(antimilitary(nixon))

A has an argument for ~(antimilitary(nixon)): republican(nixon)
B has an argument for antimilitary(nixon): quaker(nixon)
   B is CHECKING SUBARGUMENTS to block A's establishment of ~(antimilitary(nixon))
   TARGETS: ~(footballfan(nixon)) antimilitary(nixon)
   NEW TARGET for B: ~(footballfan(nixon))
   B is ESTABLISHING ~(footballfan(nixon)) for viability
   B has no argument for ~(footballfan(nixon))
   B FAILED TO ESTABLISH ~(footballfan(nixon)) for viability
   NEW TARGET for B: antimilitary(nixon)
   antimilitary(nixon) already PENDING: B backtracks
   B ALLOWS A's subarguments for ~(antimilitary(nixon))
      A is CHECKING SUBARGUMENTS to defeat B's establishment of antimilitary(nixon)
      TARGETS: ~(pacifist(nixon)) ~(antimilitary(nixon))
      NEW TARGET for A: ~(pacifist(nixon))
      A is ESTABLISHING ~(pacifist(nixon)) for support
      A has an argument for ~(pacifist(nixon)): quaker(nixon) republican(n
ixon)
         B is CHECKING SUBARGUMENTS to block A's establishment of ~(pacifist(nixon))
         TARGETS: pacifist(nixon)
         NEW TARGET for B: pacifist(nixon)
         pacifist(nixon) already PENDING: B backtracks
         B ALLOWS A's subarguments for ~(pacifist(nixon))
         B CHECKING FOR TOP-LEVEL COUNTERARGS to ~(pacifist(nixon))
         B has an argument for pacifist(nixon): quaker(nixon)
            A is CHECKING SUBARGUMENTS to defeat B's establishment of pacifist(nixon)
            TARGETS: ~(pacifist(nixon))
            NEW TARGET for A: ~(pacifist(nixon))
            ~(pacifist(nixon)) already PENDING: A backtracks
            A ALLOWS B's subarguments for pacifist(nixon)

************ round 1 ************
      A's ARGUMENT: (1)      B's ARGUMENT: (-1)
      A ACTIVATOR: (republican(nixon) ^ quaker(nixon))
      B ACTIVATOR: quaker(nixon)
      A's ARGUMENT IS MORE SPECIFIC
      B seeks another argument against ~(pacifist(nixon))
      no more arguments for B
      A ESTABLISHED ~(pacifist(nixon)) for support
      A DISALLOWS B's subarguments for antimilitary(nixon)
no more arguments for B
A WINS ~(antimilitary(nixon))
1
0.183326 seconds
```

```
INPUT:
republican(nixon) ^ quaker(nixon)!
R x republican(x) ^ quaker(x) >- ~pacifist(x).
R x quaker(x) >- pacifist(x).
R x republican(x) >- footballfan(x).
R x footballfan(x) >- ~antimilitary(x).
R x pacifist(x) >- antimilitary(x).
pacifist(nixon)?
DISPUTING pacifist(nixon)

A tries to support:  pacifist(nixon)

A has an argument for pacifist(nixon):   quaker(nixon)
B has an argument for ~(pacifist(nixon)):   quaker(nixon) republican(nixon)
   B is CHECKING SUBARGUMENTS to block A's establishment of pacifist(nixon)
   TARGETS:  ~(pacifist(nixon))
   NEW TARGET for B:  ~(pacifist(nixon))
   ~(pacifist(nixon)) already PENDING: B backtracks
   B ALLOWS A's subarguments for pacifist(nixon)
      A is CHECKING SUBARGUMENTS to defeat B's establishment of ~(pacifist(nixon))
      TARGETS:  pacifist(nixon)
      NEW TARGET for A:  pacifist(nixon)
      pacifist(nixon) already PENDING: A backtracks
      A ALLOWS B's subarguments for ~(pacifist(nixon))

************ round 1 ************
A's ARGUMENT:  (1)B's ARGUMENT:  (-1)
A ACTIVATOR: quaker(nixon)
B ACTIVATOR: (republican(nixon) ^ quaker(nixon))
B's ARGUMENT IS MORE SPECIFIC
no more arguments for A
A exhausts arguments for: pacifist(nixon)

B tries to support: ~(pacifist(nixon))

   A is CHECKING SUBARGUMENTS to block B's establishment of ~(pacifist(nixon))
   TARGETS:  pacifist(nixon)
   NEW TARGET for A:  pacifist(nixon)
   pacifist(nixon) already PENDING: A backtracks
   A ALLOWS B's subarguments for ~(pacifist(nixon))
      B is CHECKING SUBARGUMENTS to defeat A's establishment of pacifist(nixon)
      TARGETS:  ~(pacifist(nixon))
      NEW TARGET for B:  ~(pacifist(nixon))
      ~(pacifist(nixon)) already PENDING: B backtracks
      B ALLOWS A's subarguments for pacifist(nixon)

************ round 2 ************
B's ARGUMENT:  (1)A's ARGUMENT:  (-1)already argued; B won
no more arguments for A
B WINS ~(pacifist(nixon))
-1
0.166660 seconds
```