Reasoning in Attempto Controlled English: Non-Monotonicity

Norbert E. Fuchs

Department of Informatics & Institute of Computational Linguistics
University of Zurich
fuchs@ifi.uzh.ch
http://attempto.ifi.uzh.ch/

Abstract. RACE is a first-order reasoner with equality for Attempto Controlled English (ACE) that can show the consistency of a set of ACE axioms and deduce ACE theorems and ACE queries from ACE axioms. This paper presents various forms of non-monotonic reasoning.

Keywords: controlled natural language, Attempto Controlled English, ACE, RACE, monotonic reasoning, non-monotonic reasoning, abduction

1 Introduction

Attempto Controlled English (ACE)[1] is a logic-based knowledge representation language that uses the syntax of a subset of English. The Attempto Reasoner RACE[2] – one of several reasoners available for ACE[3] – allows users to show the consistency of an ACE text, to deduce one ACE text from another one, and to answer ACE queries from an ACE text. More about ACE and RACE is found in the relevant documentation[4].

A previous system description [1] of RACE detailed its structure, its functionality, its implementation and its user interfaces. The main part of [1] covered reasoning with what could be called the first-order subset of ACE, that is all ACE constructs – including alethic modality – that can be directly or indirectly mapped to first-order formulas. Furthermore, [1] described summation, a form of second-order reasoning combining the results of first-order proofs.

A closer look reveals that all reasoning examples found in [1] have one feature in common, namely that they rely on monotonic logic, i.e. adding axioms will not invalidate deductions from the original axioms.

This paper extends RACE's reasoning to non-monotonic logic[5] and presents solutions to some of its many facets. Please note that the paper is a further system description focussing on RACE's implementation. Thus you will not learn much about

[1] http://attempto.ifi.uzh.ch/
[2] http://attempto.ifi.uzh.ch/race/
[3] http://attempto.ifi.uzh.ch/site/resources/
[4] http://attempto.ifi.uzh.ch/site/docs/
[5] https://en.wikipedia.org/wiki/Non-monotonic_logic

non-monotonic reasoning in general beyond what is needed to explain the selected topics and examples.

The rest of this paper is organised as follows. In section 2, I recall general features of RACE. Section 3 recounts summation and shows that it has both monotonic and non-monotonic aspects. In section 4, I focus on variations of non-monotonic reasoning. Section 5 presents a form of abduction. Section 6 concludes with a summary of the presented solutions and with a discussion of their strengths and limitations, specifically addressing the question of their suitability.

2 General Features of RACE

For the convenience of the reader and to make this paper self-contained the material of this section is partially copied from [1].

RACE has the following general features:

- RACE offers consistency checking, textual entailment and query answering of ACE texts.
- For inconsistent ACE axioms RACE will list all minimal subsets of the ACE axioms that lead to inconsistency. If ACE axioms entail ACE theorems, respectively ACE queries, RACE will list all minimal subsets of the ACE axioms that entail the theorems, respectively queries.
- RACE is loosely based on the model-generator Satchmo [2], but offers much additional functionality.
- RACE is implemented as a set of Prolog programs that can be used locally. Furthermore, RACE can be accessed remotely via its web-client[6] or via its web-service[7].
- RACE uses about 50 auxiliary axioms to represent domain-independent general knowledge. These auxiliary axioms are not expressed in ACE, but in Prolog.
- Using the entailment A $\vdash$ T of ACE theorems T from ACE axiom A as an example, here is a sketch RACE's proof procedure.

  In a first step, the ACE axioms A and the ACE theorems T are separately translated by the Attempto Parsing Engine (APE)[8] into semantic representations called discourse representation structures DRS_A, respectively DRS_T [3].

  In a second step, (DRS_A & $\neg$ DRS_T) is translated into a set of clauses `clause(Body,Head,Index)` where `Body` is `true` or a conjunction of logical atoms and `Head` is `fail` or a disjunction of logical atoms. `Body` implies `Head`. `Index` contains the number of the ACE axiom, respectively ACE theorem, from which the clause was derived. Negation is expressed as implication to `fail`.

  In a third step, the clauses are executed bottom-up by forward-reasoning. If the `Body` of a clause is `true` or can be proved from the data base then `Head` is asserted to the database together with a list that contains `Index` and the sentence indices of all clauses that were used to prove `Body`. This amounts to building a

[6] http://attempto.ifi.uzh.ch/race/

[7] http://attempto.ifi.uzh.ch/ws/race/racews.perl

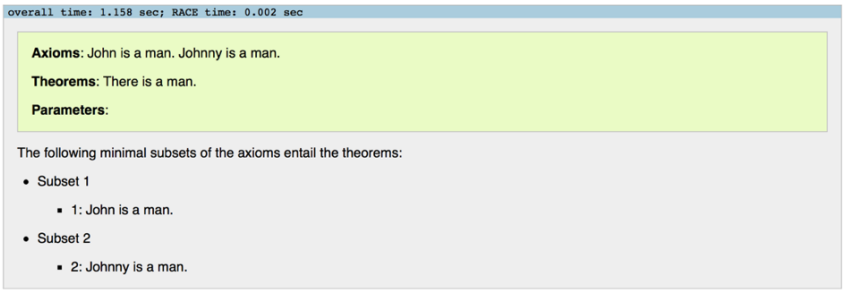
[8] http://attempto.ifi.uzh.ch/site/resources/

proof-tree labelled with lists of indices. Range restriction[9] ensures that all atoms added to the data base are ground. If Head is `fail` then the respective branch of the proof-tree is considered closed, the indices of its leaf are stored and its nodes are removed from the data base.

Forward-reasoning ends when no (further) clause can be executed. The result is either a set of ground atoms in the data base constituting a minimal Herbrand model of the clauses – indicating that the clauses are consistent, the ACE axioms do not entail the ACE theorems – or an empty data base and a set of closed branches. If all branches of the proof-tree are closed – indicating a succeeding proof– then the indices of their leaves are combined and mapped to their respective ACE axioms that constitute a minimal subset of the ACE axioms needed to entail the ACE theorems. Since RACE checks all possibilities to prove the bodies of the clauses, it can generate more than one proof.

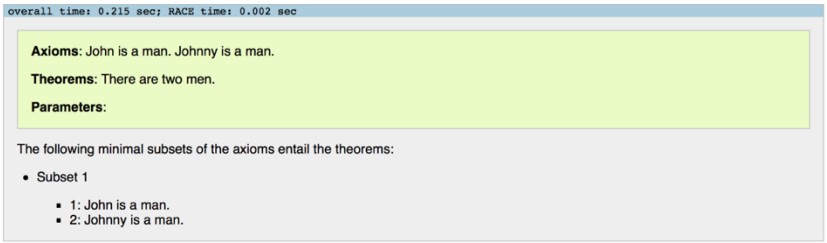
3 Monotonic and Non-Monotonic Summation

If ACE axioms entail ACE theorems RACE will list all minimal subsets of the axioms that entail the theorems. In other words, RACE will generate more than one proof of a theorem if the axioms allow for this. Here is a simple example shown as a screen-shot of the output window of RACE's web-client.



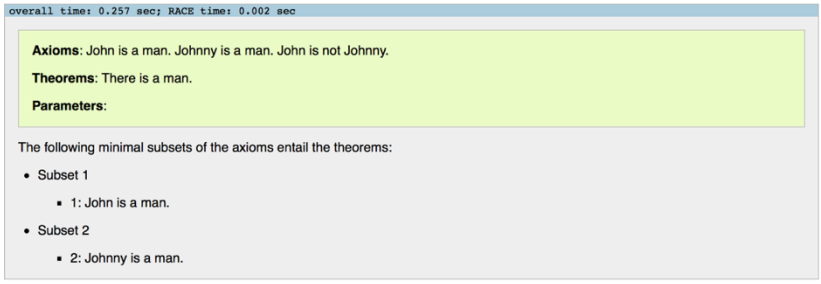
The example relies on RACE using the unique name assumption, i.e. the men *John* and *Johnny* are by default distinct. Thus there are two proofs for the theorem *There is a man*.

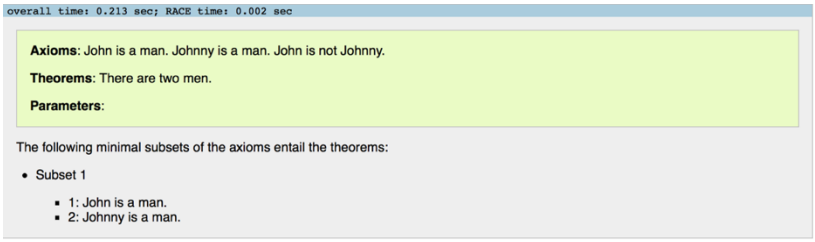
RACE provides also summation, i.e. second-order aggregation over first-order proofs. Using the same axioms and the theorem *There are two men*. we get



[9] A clause is called range-restricted if all variables of its head occur already in its body. Clauses derived from discourse representation structures are by default range-restricted.

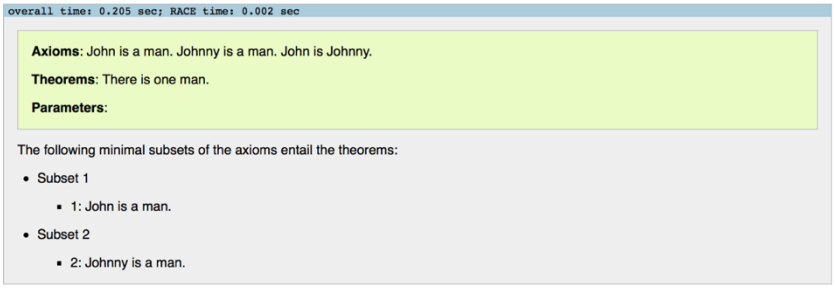
Now let's add the axiom *John is not Johnny*. and try both proofs again.

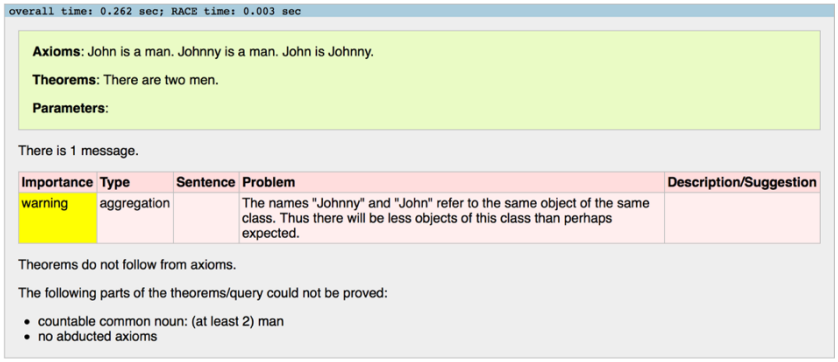




As we see, we get the same results as before. RACE behaves monotonically since the additional axiom does not prevent the deductions from the original axioms. Actually, this is no wonder since the additional axiom is just an explicit statement of the unique name assumption.

Alternatively, let's add the axiom *John is Johnny*. and try both proofs again.





While the theorem *There is a man.* can be reproduced as before, the theorem *There are two men.* cannot since the unique name assumption is explicitly overridden, and we have only one man that happens to carry two names. Thus in this case RACE behaves non-monotonically since the additional axiom prevents a deduction from the original axioms.

To sum up, RACE exhibits both monotonic and non-monotonic behaviour for summation. While this behaviour was unintended, it motivated me to extend RACE by non-monotonic reasoning in general.

4 Non-Monotonic Reasoning

Reasoning is called monotonic when derived conclusions are not invalidated by added premises. However, there are also highly relevant forms of reasoning – called non-monotonic – that derive tentative conclusions from assumed premises. Both the assumed premises and the tentative conclusions may have to be withdrawn or replaced in the light of further evidence. Non-monotonic reasoning plays an important role in everyday thinking and argumentation, in semiformal settings like medical diagnosis, or in formal domains like expert systems and physics.

Since the 1980s various formal frameworks have been developed for non-monotonic reasoning [4]. In the last years the focus has been on logic-based approaches like Answer Set Programming (ASP)[10]. Given RACE's foundation in first-order logic, it is not surprising that also its approach to non-monotonic reasoning is logic-based.

Of the many different forms of non-monotonic reasoning[11] I will consider two:

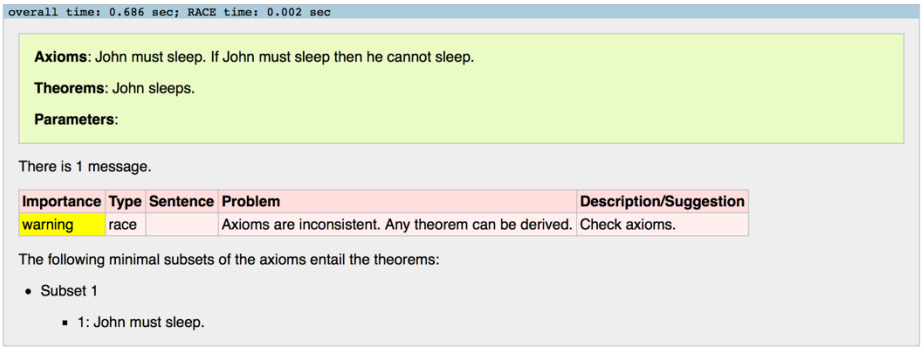
- Contradictory premises where one or the other premise may have to be withdrawn or modified to remove the contradiction; this case is traditionally not subsumed under the label "non-monotonic reasoning", but shows many similarities.
- Default reasoning where conclusions depend on default premises that express knowledge that is typically, but not necessarily always, true. Default reasoning comes in many variations some of which will be addressed.

Contradictory Axioms. The next example was specifically chosen to show that contradictory axioms are not necessarily so easily perceived that the user becomes aware of the contradiction. RACE, however, will detect it and notify the user.

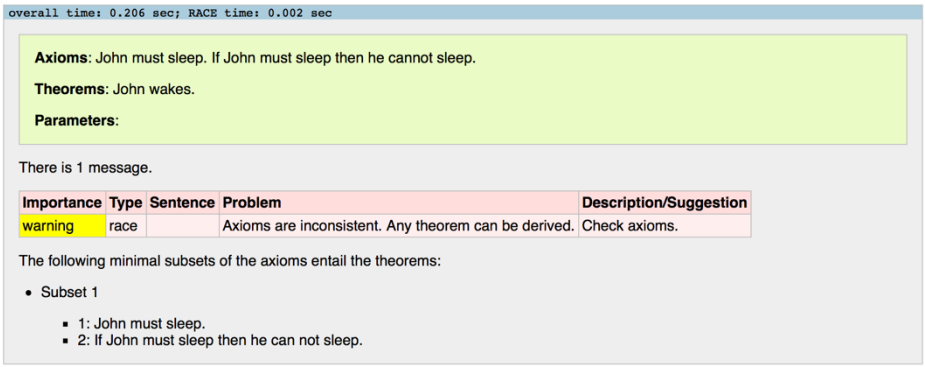
RACE handles alethic modality (necessity, possibility) by mapping modal language constructs to standard first-order logic constructs[12]. The following example makes use of the fact that the modal statement *John must sleep.* entails both the modal statement *John can sleep.* and the non-modal statement *John sleeps.* Thus the two axioms are contradictory. Notice that – as if RACE would make an effort to give the user as much information as possible – the theorem is nevertheless derived from the first axiom. More importantly, there is a warning message and the user must decide how to remove the contradiction.

---

[10] http://www.kr.tuwien.ac.at/staff/tkren/pub/2009/rw2009-asp.pdf
[11] http://plato.stanford.edu/entries/logic-nonmonotonic/
[12] https://en.wikipedia.org/wiki/Standard_translation



Worse, contradictory axioms allow users to derive a theorem – e.g. *John wakes*. – that does not follow at all from the axioms though it seems related to the subject matter.



RACE cannot prevent its users from constructing cases like this one, but alerts them by issuing a warning message, and inviting them to change the axioms.

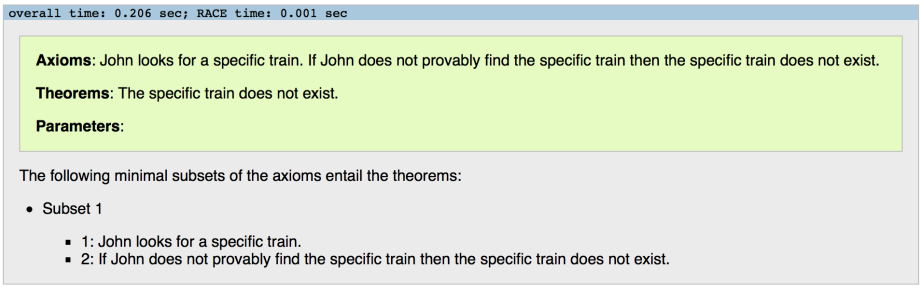
Default Reasoning: Defeasible Information. John is checking a train time-table for a specific train. Not finding this train he concludes that it does not exist. John's reasoning is based on the so-called *closed-world assumption*[13] that states that the information available is assumed complete, that everything that is true is known to be true, and that everything that is not known to be true is considered false. The *closed-world assumption* leads to defeasible conclusions, i.e. conclusions that may have to be withdrawn in the light of new evidence. When John detects that he by mistake checked an outdated time-table and that the actual time-table does contain the train he is looking for, he has to revise his previous conclusion.

In addition to logical negation ACE provides language constructs (... *does/do/is/are not provably ..., it is not provable that* ...) that stand for negation as failure (NAF)[14]. Using these language constructs together with logical negation I implemented the *closed-world assumption* in RACE. RACE allows negation as failure constructs only in the preconditions of ACE's if-then statements which leads to entries `naf(NAF)` in

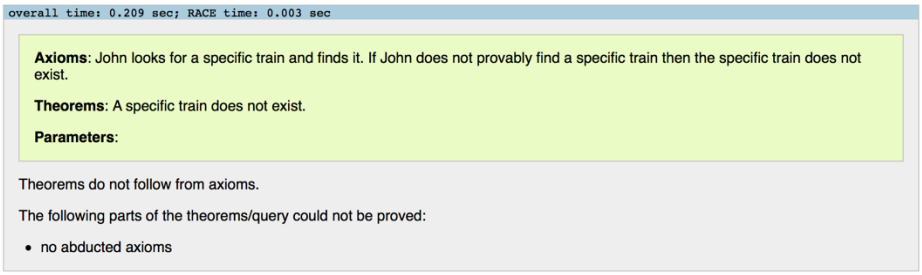
[13] https://en.wikipedia.org/wiki/Closed-world_assumption
[14] https://en.wikipedia.org/wiki/Negation_as_failure

the bodies of RACE's clauses. `NAF` is a list of logical atoms that RACE tries to prove as described in section 2. If the proof of `NAF` succeeds then `naf(NAF)` fails, and vice versa.

Here is the train example for the initial situation when John searches the outdated time-table, and – not finding the specific train – concludes that it does not exist.



If John looks for and finds the specific train in the actual time-table, he no longer deduces that the train does not exist. RACE also reproduces this behaviour.

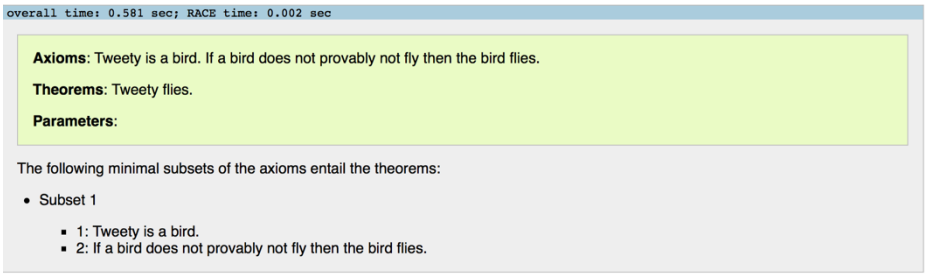


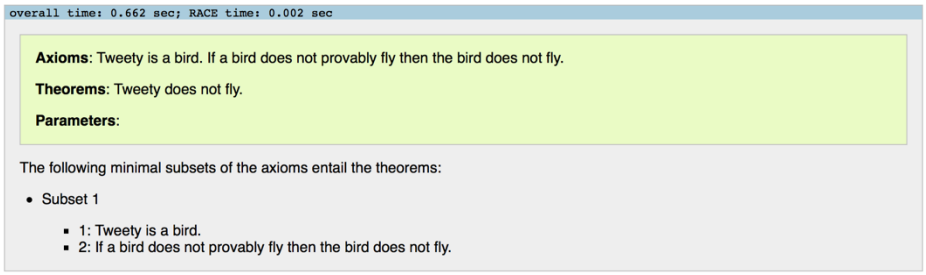
Default Reasoning: Working With Exceptions. An often-cited example of default reasoning concerns the fact that birds typically fly, but that there are exceptions like penguins that are birds, but do not fly. Thus the statement *All birds fly*. does not correctly represent reality. Replacing this statement by *All birds fly with the exception of penguins, ostriches, kiwis, dead birds, young birds, wounded birds, …* obviously is not practicable for automatic reasoning since the list of non-flying birds would have to be updated again and again.

As in the previous case negation as failure in combination with logical negation offers a convenient way to express typicality and its exceptions.

We can represent the bird example in two ways, once with a rule that birds fly if there is no evidence to the contrary, and once with a rule that birds do not fly if there is no evidence that in fact they do.

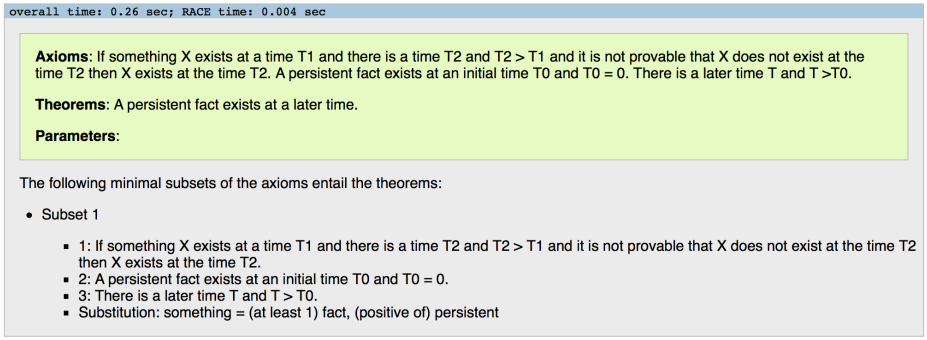
The next two figures show these two cases. Notice that in the first case the negation as failure and the logical negation occur in the precondition of the if-then statement that expresses the typicality plus exception, while in the second case the negation as failure occurs in the precondition and the logical negation in the consequence. Which formulation to choose depends among other considerations on the theorem to be proved.





Default Reasoning: Frame Problems. The frame problem[15] of artificial intelligence emerges when one parameter of a complex situation is changed and the question arises how to represent that all other parameters of the situation that are not dependent on the changed parameter stay unchanged. Of the many solutions to the frame problem those that are based on default logic[16] and on answer set programming (ASP)[17] seem to be the simplest since they directly express the common sense law of inertia, also called Leibniz law, that everything can be assumed to remain in the state in which it is.

Taking the ASP formulation of the frame axiom – *if r(X) is true at time T, and it can be assumed that r(X) remains true at time T+1, then we can conclude that r(X) remains true* – as a guidance here is an example how the frame problem can essentially be solved by RACE.

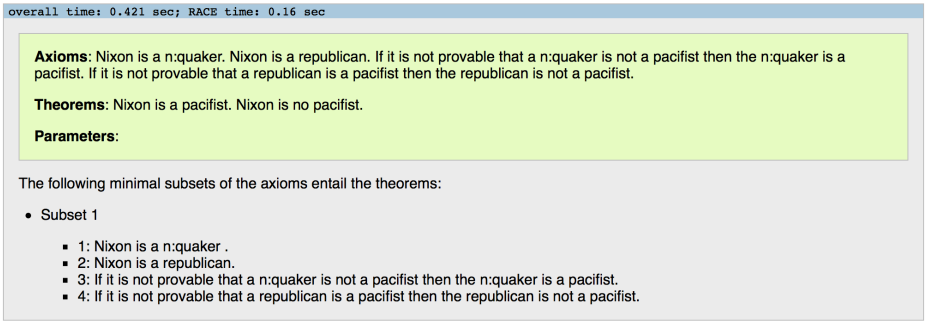


[15] https://en.wikipedia.org/wiki/Frame_problem
[16] https://en.wikipedia.org/wiki/Frame_problem#Default_logic_solution
[17] https://en.wikipedia.org/wiki/Frame_problem#Answer_set_programming_solution

The first axiom – like ASP relying on a combination of logical negation and negation as failure – expresses the persistence of situation parameters not affected by the change of another parameter. The second and third axiom describe an elementary situation.

Default Reasoning: Nixon Diamond. Default reasoning can also lead to an impasse. In the problem called Nixon diamond default assumptions lead to mutually inconsistent conclusions that require additional reasoning steps. The situation is as follows: Nixon is both a quaker and a republican. Quakers tend to be pacificists, while republicans usually do not. From these premises one can deduce that Nixon is both a pacifist and no pacifist.

Please note that in the following screen-shot the word *quaker* is prefixed by *n:* to identify it as noun since *quaker* is not found in RACE's lexicon.



To get out of this impasse there are several possibilities – all of which could in principle be implemented in RACE, but are not. First, one can assume either a skeptical attitude – the conflicting conclusions must not be drawn – or a credulous attitude – the conflicting conclusions are derived, but must be resolved in one way or other in the light of further evidence. Second, like [5] one can introduce priorities for the two default rules so that only one conclusion is derived.

5 Abduction

Abduction – another form of non-monotonic reasoning – strives to find a likely explanation for a phenomenon. In terms of reasoning this means extending given premises that do not give the desired conclusions by further premises selected from a background theory, possibly subject to constraints like simplicity or plausibility.

The question is where to find the additional premises. One answer is suggested by Abductive Logic Programming[18] that introduces logic programs with clauses whose bodies consists of so-called abducible predicates that are only partially defined and that possibly underlie a set of first-order constraints. The idea of abductive logic programming is to extend the given logic program by definitions of the abducible predicates – respecting the constraints – so that the extended program generates the

[18] https://en.wikipedia.org/wiki/Abductive_logic_programming

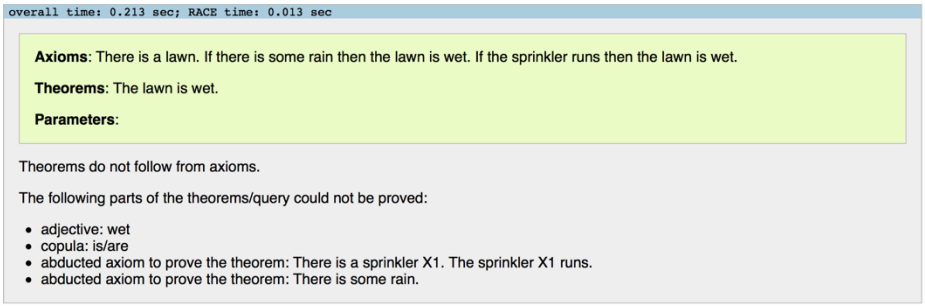
desired answer. In essence, the information that leads to abducted premises is contained in the bodies of the given clauses.

This approach inspired RACE's strategy for a form of abduction:

- axioms contain if-then statements whose preconditions are incompletely defined; possibly there are some constraints for these preconditions
- theorems are matched against the conclusions of the if-then statements; matching may have to extend to all given axioms if noun phrases occur in the conclusions only as anaphoric references, but are actually defined elsewhere
- if matching succeeds then the precondition of the respective if-then statement is added to the axioms respecting the constraints and avoiding inconsistency and duplication

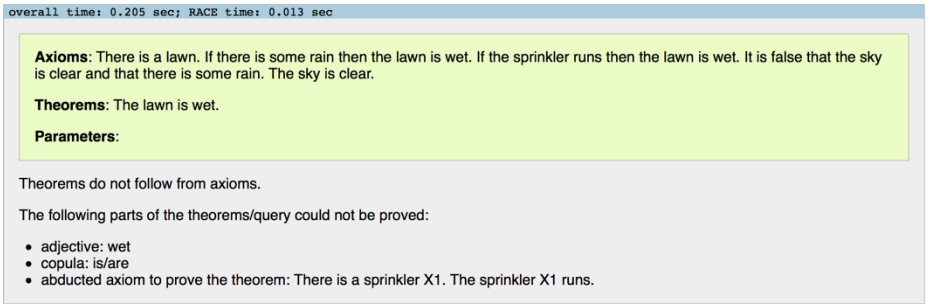
Here is a simple example[19]: If your lawn is wet then it either rained or you had switched on the sprinkler. If the sky is clear you must exclude rain as the cause, so that the sprinkler is the cause.

First the example without the constraint.



We get two sets of abducted axioms that each – together with the original axioms – prove the theorem.

Now we add the constraint that it cannot rain from a clear sky and the fact that the sky is clear.



As expected we get only one set of abducted axioms that together with the original axioms prove the theorem.

[19] https://en.wikipedia.org/wiki/Abductive_logic_programming#Example_1

6 Conclusions

I extended the Attempto Reasoner RACE by non-monotonic reasoning using small examples that focus on the respective issues.

As described in [1] RACE relies on auxiliary axioms that add domain-independent general knowledge to the domain-specific knowledge of the given axioms. Since these auxiliary axioms are coded in Prolog that has the power of the Turing machine, RACE could in principle deduce any conclusion from the axioms. As a consequence – in addition to the usual questions of correctness and completeness of a theorem prover – a further question arises, namely what RACE should actually deduce. The answer to this question depends on the domain investigated and on the expectations and intuitions of the users, and – as my experience has shown – may be highly debatable.

In non-monotonic reasoning a similar question of suitability arises, this time related to the reasoning methods I chose. While default reasoning based on ACE's language constructs negation and negation as failure seems quite general and powerful, the same cannot yet be said for RACE's strategy for abduction inspired by abductive logic programming. Thus questions arise, for example which of my methods is dictated by the example problems, and which is powerful and general enough to be applied to other types of problems. In other words, the methods and implementations of non-monotonic reasoning that I presented in this paper need further investigations.

RACE now covers all language constructs of Attempto Controlled English with the exception of those that have no direct logical representations – imperative sentences and the modal operators for recommendation (*should*) and admissibility (*may*) – and the operations on lists, sets and strings that – not posing any problems as far as reasoning is concerned – will be implemented in RACE at some other time.

Acknowledgements

I would like to thank the three anonymous reviewers of the first version of this paper for their constructive comments. Many thanks go to the Department of Informatics and the Institute of Computational Linguistics, University of Zurich, for their hospitality.

References

1. Fuchs, N. E.: First-Order Reasoning for Attempto Controlled English, Proc. Second International Workshop on Controlled Natural Language (CNL 2010), Springer (2012)
2. Manthey, R., Bry, F.: SATCHMO: A Theorem Prover Implemented in Prolog. In: Lusk, E. L., Overbeek, R. A. (eds.). Proc. CADE 88, LNCS 310, pp. 415—434. Springer (1987)
3. Bos, J.: Computational Semantics in Discourse: Underspecification, Resolution, and Inference. Journal of Logic, Language and Information, vol. 13, pp. 139—157 (2004) ; Fuchs, N. E., Kaljurand, K., Kuhn, T.: Discourse Representation Structures for ACE 6.6, Technical Report ifi-2010.0010, Department of Informatics, University of Zurich (2010)
4. Genesereth, M. R., Nilsson N. J.: Logical Foundations of Artificial Intelligence, Morgan Kaufmann Publishers (1987)
5. Grosof, B. N.; Courteous Logic Programs: Prioritized Conflict Handling For Rules. IBM Research Report RC 20836. IBM T. J. Watson Research Center (1997)