I N F S Y S
R E S E A R C H
R E P O R T

INSTITUT FÜR INFORMATIONSSYSTEME

ABTEILUNG WISSENSBASIERTE SYSTEME

Heterogeneous Active Agents, II: Algorithms and Complexity

Thomas Eiter V.S. Subrahmanian

INFSYS RESEARCH REPORT 1843-98-03

DECEMBER 1998

Institut für Informationssysteme
Abtg. Wissensbasierte Systeme
Technische Universität Wien
Treitlstraße 3
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at

TECHNISCHE UNIVERSITÄT WIEN

INFSYS RESEARCH REPORT

INFSYS RESEARCH REPORT 1843-98-03, DECEMBER 1998

Heterogeneous Active Agents, II: Algorithms and Complexity

Thomas Eiter[1] V.S. Subrahmanian[2]

Abstract. In Part I of this series of papers, we developed a language called *Agent Programs* for defining the operational behavior of software agents and defined a set of successively more satisfying (epistemically) semantics for such agent programs. In Part II of this series of papers, we study the computation price to be paid (in terms of complexity) for these epistemic desiderata. In particular, we develop algorithms for the above semantics, and describe results on their computational complexity. We show that (surprisingly) the reasonable status set semantics is the easiest to compute of the semantics proposed.

[1]Institut und Ludwig Wittgenstein Labor für Informationssysteme, Technische Universität Wien, Paniglgasse 16, A-1040 Wien, Austria. E-mail: eiter@kr.tuwien.ac.at

[2]Institute for Advanced Computer Studies, Institute for Systems Research and Department of Computer Science, University of Maryland, College Park, Maryland 20742. E-mail: vs@cs.umd.edu

Acknowledgements: We wish to thank Alex Dekhtyar, Jürgen Dix, Sarit Kraus, Munindar Singh, Terrence Swift, and others for a very close reading of previous versions of this manuscript, and for the numerous detailed comments they made. These comments have significantly improved the quality of this paper. We have also benefited from discussions with Swati Allen, Piero Bonatti, Steve Choy, Phil Emmerman, Dana Nau, Anil Nerode, Dino Pedreschi, and Jose Salinas.
This work was supported by the Army Research Office under Grants DAAH-04-95-10174, DAAH-04-96-10297, and DAAH04-96-1-0398, by the Army Research Laboratory under contract number DAAL01-97-K0135, by an NSF Young Investigator award IRI-93-57756, and by a DAAD grant. his work was initiated by R. Stewart.

This is Part II of a revised and extended version of the Technical Report UMCP-CSD CS-TR-3880, “Heterogeneous Active Agents”, Computer Science Department, University of Maryland.

Contents

1 Introduction **1**

2 Algorithms and Complexity Issues **2**
2.1 Underlying assumptions . . . 2
2.2 Brief overview of complexity classes . . . 3
2.2.1 Decision problems and search problems . . . 3
2.2.2 Selected complexity classes . . . 4
2.2.3 Hardness and completeness . . . 6
2.3 Brief overview of complexity results . . . 6
2.3.1 Bottom line for the computation problem . . . 8
2.3.2 Sources of Complexity . . . 10

3 Complexity Results for the Case Without Integrity Constraints **11**
3.1 Positive programs . . . 12
3.1.1 Weak rational status sets . . . 14
3.2 Programs with negation . . . 17
3.2.1 Feasible status sets . . . 17
3.2.2 Rational status sets . . . 18
3.2.3 Reasonable status sets . . . 22
3.2.4 Weak status sets . . . 23
3.2.5 Preferred status sets . . . 28

4 Complexity Results for the Case With Integrity Constraints **32**
4.1 Feasible status sets . . . 32
4.2 Rational status sets . . . 34
4.3 Reasonable status sets . . . 36
4.4 Weak status sets . . . 36
4.4.1 Positive programs . . . 36
4.4.2 Programs with negation . . . 40
4.5 Preferred status sets . . . 43

5 Conclusion **46**

References **48**

A Appendix: SAT Problems and Quantified Boolean Formulas **49**

B Appendix: Notation and Different Kinds of Status Set from Part I **50**

C Appendix: Proof of Lemma 4.16 **52**

1 Introduction

In Part I of this series of papers, we have defined an architecture for the creation and deployment of software agents — our platform supports building such agents both from scratch, and extending existing legacy applications to handle such agent capabilities. Our architecture of an agent consists of five basic parts.

- A description of the set of data types the agent manages, together with the function calls it uses to manipulate these types. The state of the agent at a given point in time is the set of objects belonging to these data types that is currently resident in the agent's working memory.
- A set called the *action base* of the agent consisting of *actions* that the agent is physically capable of taking — these actions alter the state of the agent and may be viewed as *transactions* (in the sense of databases and operating systems) [25].
- A notion of *concurrency* which specifies what it means to execute certain actions concurrently.
- A language called *agent programs* through which, the agent's designer specifies the operating principles (what actions the agent must do, what actions the agent may do, what actions the agent may not do, etc.) of the agent.
- A set of *integrity constraints* that the agent's state must always satisfy. In particular, when a set of actions is executed by the agent in a state that satisfies the integrity constraints, then the new state that results must also satisfy the integrity constraints.
- A set of *action constraints* that specifies the circumstances under which certain actions may be concurrently executed.

In Part I of this series of papers, we characterized the semantics of an agent program through the notion of a *status set*. Intuitively, a *status set* is a set of *status atoms* which are formulas of the form $Op\alpha$ where α is the name of an action, and Op is a modality $\mathbf{P}, \mathbf{O}, \mathbf{F}, \mathbf{Do}, \mathbf{W}$. Intuitively, $\mathbf{P}\alpha$ means α is permitted, $\mathbf{F}\alpha$ means α is forbidden, $\mathbf{O}\alpha$ means α is obligatory, $\mathbf{Do}\alpha$ means α is done, and $\mathbf{O}\alpha$ means that the obligation to do α is waived. The main idea in Part I of this series of papers was that at time t, the agent's previous state $\mathcal{O}_{t-1}$ changes through the receipt of one or more messages. The agent must compute an "appropriate" status set S_t and concurrently perform all actions of the form $\mathbf{Do}\alpha$ in S_t so as to transit to a new state, $\mathcal{O}_t$. Part I of this series of papers describes several ways of capturing the word "appropriate" used in the previous sentence. Each of these ways yields a different semantics for agent programs. Part I of this series of papers shows that these different semantics appeal to different epistemic intuitions that an agent developer must have — some are epistemically more desirable than others.

The main aim of this paper is to analyze the computational complexity of the above semantics so that we have a clear idea of the computation price being paid (if any) for an epistemically desirable semantics. A further consequence of these complexity results is that we are able to pinpoint correct algorithms to compute these different semantics.

The organization of this paper is as follows. Section 2 specifies the assumptions underlying our complexity analysis. It also provides a brief tutorial of different complexity classes, and then provides a succinct summary of all the complexity results derived in this paper.

The main complexity results and accompanying algorithms are contained in Sections 3 and 4 — the former contains results when no integrity constraints are present, and the latter contains results when integrity constraints may be present. Each of these sections is further broken down in two two parts — when no negation may appear in the body of an agent program rule, and when such negations can appear. Finally, Section 5 concludes the paper. As a handy reference for the reader, some notation and definitions from Part I are provided in Appendix B.

2 Algorithms and Complexity Issues

We assume that the reader is familiar with the basic concepts of complexity theory, in particular with NP-completeness and the polynomial hierarchy, and refer to [12, 18, 20] for background material on this subject and for concepts and notation that we use in the remainder of this paper.

In the rest of this section, we first present the assumptions we make for our analysis. We then present a very brief tutorial on different complexity classes. Finally, we present an overview and a discussion of the results that we derive. The reader who is interested in algorithms derived from these results, and/or the formal proofs of the results will find them in Sections 3 and 4.

2.1 Underlying assumptions

In our work, we consider the evaluation of a fixed agent program $\mathcal{P}$ in the context of software code $\mathcal{S}$, an action base $\mathcal{AB}$, action constraints $\mathcal{AC}$, and integrity constraints $\mathcal{IC}$, each of which is fixed, over varying states $\mathcal{O}_{\mathcal{S}}$. This corresponds to what researchers in databases and logic programming commonly call the *data complexity* of a program [26]. If we consider varying programs where the agent state is fixed (resp., varying), we would have *expression (or program) complexity and combined complexity*, which are typically one exponential higher than data complexity. This also applies in many cases to the results that we derive below; such results can be established using the complexity upgrading techniques for expression complexity described in [14].

Of course, if we use software packages $\mathcal{S} = (\mathcal{T}_{\mathcal{S}}, \mathcal{F}_{\mathcal{S}})$ with high intrinsic complexity, then the evaluation of agent programs will also be time consuming, and leaves us no chance to build efficient algorithms. We therefore have to make some general assumptions about the software package used such that polynomial time algorithms are not a priori excluded.

Domain Closure Assumption: We adopt a *generalized active domain* assumption on objects, in the spirit of domain closure; all objects considered for grounding the program rules, evaluation of the action preconditions, the conditions of the actions constraints and the integrity constraints must be from $\mathcal{O}_{\mathcal{S}}$, or they must be constructible from objects therein by operations from a fixed (finite) set in a number of steps which is bounded by some constant, and such that each operation is efficiently executable (i.e., in polynomial time) and involves only a number of objects bounded by some other constant. Notice that the active domain assumption is often applied in the domain of relational databases, and a similar domain closure assumption [22] is frequently made in the context of knowledge bases. In our framework, creation and use of tuples of bounded arity from values existing in a database would be a feasible object construction process, while creation of an arbitrary relation (as an object that amounts to a set of tuples) would be not.

Under this assumption, the number of objects which may be relevant to a fixed agent program $\mathcal{P}$ on a given state $\mathcal{O}_{\mathcal{S}}$ is bounded by a polynomial in the number of objects in $\mathcal{O}_{\mathcal{S}}$, and each such object can be generated in polynomial time. In particular, this also means that the number of ground rules of $\mathcal{P}$ which are relevant is polynomial in the size of $\mathcal{O}_{\mathcal{S}}$, measured by the number of objects that it contains.

Polynomial Code Calls: As our framework builds on top of an existing body of software code, we will state all our results under the assumption that the evaluation time of code condition calls χ over a state $\mathcal{O}_{\mathcal{S}}$, for any particular legal assignment of objects, is bounded by a polynomial in the size of $\mathcal{O}_{\mathcal{S}}$. Moreover, we assume that given an agent state $\mathcal{O}_{\mathcal{S}}$ and a set of ground actions $\mathcal{A}$, the state $\mathcal{O}'_{\mathcal{S}}$ which results under concurrent execution of $\mathcal{A}$ on $\mathcal{O}_{\mathcal{S}}$ is constructible in polynomial time (Part I of this series of papers provides definitions of concurrency that preserve this property).

As a consequence of these assumptions, the action and integrity constraints are evaluable on an agent state $\mathcal{O}_{\mathcal{S}}$ under the generalized active domain semantics in polynomial time, and the integrity constraints on the agent state $\mathcal{O}'_{\mathcal{S}}$ resulting from the execution of a set of actions A grounded in the active domain, are checkable in polynomial time in the size of $\mathcal{O}_{\mathcal{S}}$.[1]

Notice that these assumptions will be met in many software packages which support the use of integrity constraints (e.g., a relational database). If evaluation of the code condition calls or constraints were not polynomial, then the evaluation of the agent program would not be either.

2.2 Brief overview of complexity classes

In this subsection, we present a brief tutorial on complexity theory and also briefly describe the various complexity classes that we will encounter in this paper. The classes that we use in our characterizations are summarized in Figure 1. An edge directed from class C_1 to class C_2 indicates that all problems in C_1 can be efficiently transformed into some problem in C_2, and that it is strongly believed that a reduction in the other direction is not possible; i.e., the hardest problems in C_2 are more difficult than the problems in C_1.

2.2.1 Decision problems and search problems

All computation problems involved with computing different kinds of status sets associated with agent programs are either *decision problems* or *search problems*. These two types of problems are briefly described below.

Decision Problems. Figure 1(a) shows the complexity hierarchy for *decision problems* — these are problems where a question is posed, and a "yes/no" answer is expected. Thus, problems like SAT are decision problems — SAT, for instance, asks if there is a valuation that satisfies a set of propositional clauses. Similarly, the question "Does agent program P have a feasible status set w.r.t. some fixed agent state, integrity constraints, and action constraints?" is a decision problem.

We will assume that all readers know what the classes P *and* NP *are.* The other classes shown in Figure 1(a) are built on top of the classes P and NP (which is also referred to as Σ_1^P), by allowing the use of an oracle (i.e., a subprogram) for deciding problems instantaneously. The class C to which this oracle must belong is denoted in a superscript; e.g., P^{NP} (resp., $\mathrm{NP}^{\mathrm{NP}}$) is the class of problems solvable in polynomial time on a deterministic (resp., nondeterministic) Turing machine, if an oracle for a problem in NP may be used. Similarly, the class $\mathrm{P}^{\Sigma_2^P}$ is the class of problems solvable in polynomial time on a deterministic Turing machine if an oracle for a problem in Σ_2^P may be used. In general, $C_1^{C_2}$ refers to the set of all problems that are in complexity class C_1 if we assume that there is an oracle for all problems in class C_2 that is capable of responding instantaneously. The classes Σ_i^P, Π_i^P, and Δ_1^P, where $i \geq 1$, constitute the so called *polynomial hierarchy*, which contains problems of increasing complexity, but they are supposed to be easier than PSPACE-complete problems.

For the decision classes, the arcs in Figure 1 actually denote inclusions, i.e., the transformation of problems in C_1 to problems in C_2 is by means of the identity.

[1]This would remain true if the integrity constraints where arbitrary fixed first-order formulas (evaluated under active domain semantics).

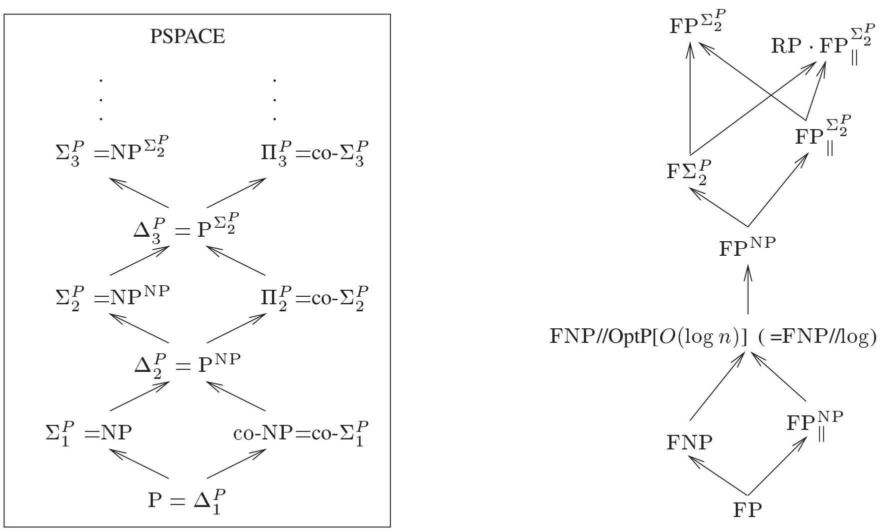

Figure 1: (a) Decision complexity classes (left) (b) Search complexity classes (right)

Search Problems. The classes for search problems, which are often also called function classes, can be found in [20, 4] (see also [23, 17]). A search problem is a generalization of a decision problem, in which for every instance I of the problem a (possibly empty) finite set $S(I)$ of solutions exists. To solve such a problem, a (possibly nondeterministic) algorithm must compute the solutions of this set in its computation branches, if it is not empty. Thus, while the decision problem SAT asks whether a set of propositional clauses is satisfiable, the corresponding search problem FSAT attempts to find a satisfying valuation if one exists. Analogously, the question "Find a feasible status set of P if one exists" is a search problem. Decision problems can be viewed as particular search problems, in which the solution set is either empty or the singleton set $\{\text{yes}\}$. Hence, decision problems are somewhat simpler than search problems.

More formally, search problems in the classes from Figure 1 are solved by transducers, i.e., Turing machines equipped with an output tape. If the machine halts in an accepting state, then the contents of the output tape is the result of the computation. Observe that a nondeterministic machine computes a (partial) multi-valued function. Thus, not all arcs in Figure 1 mean inclusion, i.e., trivial reducibility by the identity. However, if we are only interested in *some arbitrary* solution from a set of possible solutions, as e.g. in some arbitrary satisfying assingment in case of problem FSAT, then we may give up the implicit uniformity condition of having *each* solution as a possible outcomes of a (nondeterministic) computation, and simple require that at *least one* of the possible solutions is returned over all branches – this is the (natural) view that we will adopt when classifying problems on agent programs. Observe that this view, adopted also e.g. [4] for solving optimization problems, is coherent with the notion of reduction introduced in Subsection 2.2.3, and turns the arcs in Figure 1 into inclusions. For example, FSAT and finding some arbitrary feasible status set are problems in $\mathrm{FP}^{\mathrm{NP}}$ under this view.

2.2.2 Selected complexity classes

In this section, we present a few selected search complexity classes that will crop up in our complexity analysis of agent programs.

First, we note that the search problem counterparts of the classes C in the polynomial hierarchy are often denoted by a prefixed "F"; some of them appear in Figure 1.

The Classes FP**,** $\mathrm{FP}^{\mathrm{NP}}$**, and** $\mathrm{FP}^{\Sigma_2^P}$**:** These are the classes of functions computable by a deterministic Turing machine in polynomial time with no oracle, NP-oracle, and Σ_2^P-oracle, respectively. Notice that each such machine computes a single-valued function. The classes $\mathrm{FP}_{\|}^{\mathrm{NP}}$ and $\mathrm{FP}_{\|}^{\Sigma_2^P}$ are refinements of the classes $\mathrm{FP}^{\mathrm{NP}}$ and $\mathrm{F}\Sigma_2^P$, respectively, and are the search problem counterparts of the classes $\mathrm{P}_{\|}^{\mathrm{NP}}$ and $\mathrm{P}_{\|}^{\Sigma_2^P}$, respectively, which are not shown in the figure. These classes contain functions which are computable in polynomial time on a deterministic Turing machine which has access to an oracle in NP (resp., Σ_2^P), but where all queries to the oracle must be prepared before issuing the first oracle call. Thus, the oracle calls are nonadaptive and must essentially take place in parallel; it is commonly believed that this restricts computational power.

The Classes FNP **and** $\mathrm{F}\Sigma_2^P$**:** FNP (resp., $\mathrm{F}\Sigma_2^P$) contains the multi-valued functions whose solutions can be computed by a nondeterministic transducer in polynomial time (resp., in polynomial time with an NP-oracle), such that a given solution candidate can be checked in polynomial time (resp., the check is in co-NP). The class is contained in the class NPMV (resp., $\mathrm{NPMV}^{\mathrm{NP}}$), which contains all multi-valued functions computable in nondeterministic polynomial time (resp., in nondeterministic polynomial time with a NP oracle) [11].

The Class $\mathrm{FNP}//\log$**:** $\mathrm{FNP}//\log$ is short for the class FNP//OptP[$O(\log n)$] [4]. Intuitively, it is the class of problems such that a solution for an instance I can be nondeterministically computed by a transducer in polynomial time, if the optimal value $opt(I)$ of an NP optimization problem on I (an integer) is known, where $opt(I)$ (represented in binary) must have $O(\log |I|)$ bits. NP *optimization problem* means here that the maximal (resp., minimal) value of a solution for a problem Π is computed such that, given I and an integer k, deciding whether $opt(I) \geq k$ (resp., $opt(I) \leq k$) is in NP and recognizing solutions is polynomial.

E.g., computing the largest set S of pairwise connected nodes in a given graph G (i.e., a maximum clique) is a problem in $\mathrm{FNP}//\log$ (observe that different maximum cliques may exist). Indeed, computing the *size* of a maximum clique is an NP-optimization problem with $O(\log |I|)$ output bits, since testing whether a set S is a clique is easy (just check whether G has an edge between each pair of nodes in S), and deciding whether $opt(G) \geq k$ is in NP (guess a clique of size $\geq k$). Furthermore, if $s = opt(G)$ is known, then the transducer can nondeterministically generate and verify a clique of size s in polynomial time.

The class $\mathrm{FNP}//\log$ reduces to $\mathrm{FP}^{\mathrm{NP}}$ and roughly amounts to a randomized version of $\mathrm{FP}_{\|}^{\mathrm{NP}}$. Due to its nondeterministic nature, it contains problems which are not known to be solvable in $\mathrm{FP}_{\|}^{\mathrm{NP}}$. The most prominent of these problems is the computation of an arbitrary model of a propositional formula [17], which is the prototypical problem complete for the class FNP. Few natural $\mathrm{FNP}//\log$-complete problems are known to date and almost none arise in practical applications; our analysis shows that certain problems arising naturally in agent systems (e.g. computing a weak rational status set) are in $\mathrm{FNP}//\log$, and that some of them are even complete for this class.

In the context of agent programs, computing a weak rational status set for a positive program is in $\mathrm{FNP}//\log$, since if we know the maximum size $smax$ of a set A of ground actions such that an A-rational (i.e., obeying obligations according to A) status set S exists, then we can nondeterministically generate such an S in polynomial time. The computation of $smax$ amounts to an NP optimization problem as described above, and thus the overall algorithm places the problem in FNP//$\log$.

The Class $\mathrm{RP} \cdot \mathrm{FP}_{\|}^{\Sigma_2^P}$ **:** The class $\mathrm{RP} \cdot \mathrm{FP}_{\|}^{\Sigma_2^P}$ [4] contains informally those problems for which a solution on input I can be found by a random polynomial time algorithm with very high probability, by using a problem in $\mathrm{FP}_{\|}^{\Sigma_2^P}$ as single-call subroutine. This class is above $\mathrm{FP}_{\|}^{\Sigma_2^P}$. Chen and Toda [4] have shown that many optimization problems belong to this class whose solutions are the maximal (resp., minimal) solutions of an associated decision problem for which recognizing a solution is in co-NP. We shall use this relationship for classifying some problem into $\mathrm{RP} \cdot \mathrm{FP}_{\parallel}^{\Sigma_2^P}$, and refer the interested reader to [4] for the technical details about this class. As we shall see, computing an F-preferred rational status set or a weak rational status set amount to such optimization problems, and thus the problems belong to $\mathrm{RP} \cdot \mathrm{FP}_{\parallel}^{\Sigma_2^P}$.

2.2.3 Hardness and completeness

The reader would have frequently heard terms such as NP-hard, NP-complete, Σ_2^P-complete and so on. Here, NP and Σ_2^P are classes of problems. In this section, we will briefly explain what it means for a problem to be *hard/complete* w.r.t. a class of problems. The first concept we need here is that of a reduction between two problems.

Reductions: Consider two search problems Π_1, Π_2. (For example, Π_1 may be SAT, while Π_2 may be 3SAT). In general, when we say that Π_1 is reducible to Π_2, it informally means that there is a function which transforms all instances of problem Π_1 to "equivalent" instances of Π_2. Furthermore, this function is polynomially computable. Intuitively, reductions satisfy the following condition. If Π_1 is reducible to Π_2, then given any instance I of Π_1, we can transform this instance to an equivalent instance of Π_2, execute a known algorithm for Π_2, and then transform any solution for Π_2 into a solution for Π_1.

Formally, Π_1 is *polynomial-time reducible* to Π_2, if (i) from every instance I of Π_1, an instance $f(I)$ of Π_2 is constructible in polynomial time, such that $f(I)$ has some solution precisely if I has; and (ii) from every solution S of $f(I)$, a solution $g(I, S)$ of I can be constructed in time polynomial in the size of S and I. The pair of functions (f, g) constitutes a polynomial-time reduction of Π_1 to Π_2.

It is easy to see that the concept of polynomial-time reduction among decision problems Π_1 and Π_2 is a special case of this definition, because decision problems are special cases of search problems. Other types of reductions among decision problems that change the polynomial-time requirement, e.g., reduction in logarithmic space, can be generalized to search problems in the same way.

We are now ready to explain the concepts of hardness and completeness.

Hardness and Completeness: For both decision and search problem classes C, a problem Π is complete for C, if (1) Π belongs to C, and (2) Π is hard for C, i.e., every problem in C polynomially reduces to it.

Intuitively, when we say that problem Π_1 is NP-hard (or class C-hard) we mean that every problem in the class NP (resp. class C) can be reduced to Π_1 in polynomial time. Likewise, when we say Π_1 is NP-complete (or class C-complete) we mean not only that Π_1 is NP-hard (or class C-hard), but also belongs to that class, i.e. it is not strictly harder. Examples of complete problems for the complexity classes that we encounter are given in Appendix A.

2.3 Brief overview of complexity results

The complexity results we derive may be broken up into two parts. In the first part (Section 3) we assume that no integrity constraints are present — then in Section 4, we allow integrity constraints to be present.

In this paper, we study four types of complexity problems. For each semantics introduced in the paper, we study the complexity of these four problems. This leads to Tables 1 and 3 which summarize the results, under different assumptions on the syntax of the agent programs considered. Table 2 specifies where the proofs of the results listed in Table 1 and Table 4 does the same for the results listed in Table 3.

The computational problems that we study are listed below; let Sem be any kind of status sets.

(consistency) Deciding the consistency of the program on a given agent state, i.e., existence of a Sem-status set;

(recognition) the recognition of a Sem-status set;

(computation) the computation of an arbitrary Sem-status set; and

(action reasoning) reasoning about whether the agent takes an action α under Sem-status sets, both under the

- possibility variant (decide whether α is executed according to some Sem-status set), and the
- certainty variant (decide whether α is executed according to every Sem-status set).

It is easy to see that "computation" is a search problem, while the other three are decision problems (all instances of these problems can be answered with a "yes" or a "no"). Thus, the only column in Tables 1 and 3 which use the search problem hierarchy is the "computation" column.

The consistency problem is important since in general, it is not a priori guaranteed that the agent can figure out what to do by selecting some Sem-status set. It might be the case that no such status set exists. Intuitively, this means that the behavior as specified by the agent program is incompatible with the agent's state. This event causes an exception, which must be appropriately handled – this, however, is beyond the scope of this paper.

Computing some Sem-status set is closely related to the consistency problem. Of course, the computation problem is at least as difficult as the consistency problem – having a Sem-status set at hand, it is trivial to answer whether some Sem-status set exists. On the other hand, like with many other problems, computing some Sem-status set is not much harder than the decision problem, in the sense that it is possible in polynomial time with an oracle for the consistency problem. However, this does not tell us much about how an optimal (possibly nondeterministic) algorithm can proceed. For this purpose, a complexity characterisation referring to search problem classes is useful. We will come back to this issue in the conclusions (Section 5).

The recognition problem corresponds to the task of model checking in the area of knowledge representation and reasoning, which has been addressed e.g. in [15, 2, 19]. Observe that often, recognizing a solution is easier than computing a solution, and occurs as a test in an interactive algorithm. However, in general, it may be the case that recognizing a particular solution is much harder than computing some arbitrary solution. Thus, the complexities of "computation" and "recognition" are incomparable in general.

Action reasoning is a problem of interest, since in general multiple Sem-status sets may exist, and thus it is important to know whether some action status atom A belongs to all (resp., some) Sem-status set. This corresponds to what is known as certainty (resp., possibility) reasoning in databases [25], and to cautious (resp., brave) reasoning in the area of knowledge representation [13]. In particular, this question is important for status atoms $\mathbf{Do}(\alpha)$, since it tells us whether α is possibly executed by the agent (if she picks nondeterministically some Sem-status set), or executed for sure (regardless of which action set is chosen).

Table 1 specifies the complexity of the four problems that we study when positive agent programs are considered, while Table 3 specifies their complexity when arbitrary agent programs are considered.

Note on Tables: The entries for decision problems in Tables 1 and 3 stand for completeness for the respective complexity classes. In case of P, hardness may implicitly be present with costly object construction operations. However, we remark that for all problems except recognition of a feasible status set, hardness holds even if no new objects are introduced and the agent state consists merely of a relational database.

				action reasoning	
$\mathcal{IC} = \emptyset \,\vert\, \mathcal{IC}$ *arbitrary*	*consistency*	*computation*	*recognition*	*possible*	*certain*
feasible	P $\vert$ NP	FP$\vert$ FNP	P	NP	co-NP
rational $\equiv$ *reasonable* $\equiv$ F*-pref. rational* $\equiv$ F*-pref. reasonable*	P	FP	P	P	P
weak rational $\equiv$ *weak reasonable*	P $\vert$ NP	FP$\vert$ FNP//log *	P $\vert$ co-NP	NP	co-NP $\vert$ Π_2^P

* ... hard for both FNP and $\mathrm{FP}_{\Vert}^{\mathrm{NP}}$

Table 1: Complexity of fixed positive agent programs

				action reasoning	
$\mathcal{IC} = \emptyset \,\vert\, \mathcal{IC}$ *arbitrary*	*consistency*	*computation*	*recognition*	*possible*	*certain*
feasible	T 3.1 $\vert$ T 4.1	T 3.1 $\vert$ T 4.1	P 3.7	T 3.9	T 3.9
rational	T 3.1	T 3.1	C 3.2	C 3.3	C 3.3
weak rational	T 3.5 $\vert$ T 4.7	T 3.5 $\vert$ T 4.8	T 3.4 $\vert$ T 4.6	T 3.6,T 4.9	T 3.6 $\vert$ T 4.9

Table 2: Location of proofs for Table 1 (C= Corollary, T= Theorem, P= Proposition).

Proofs of these results are not difficult, using the well-known result that inference from a datalog program (Horn logic program) is P-complete, cf. [6].

2.3.1 Bottom line for the computation problem

Of all the four problems described above, from the point of view of the IMPACT system (and in general, for any system that attempts to determine which actions an agent must take), the most important problem, by far, is the problem of *computation* – given an agent program, a current agent state, a set of integrity constraints and action constraints, determine a set of actions that the agent must take. This task forms the single most important task that an agent must take, over and over again.

When considering the different semantics for agent-programs, we easily notice (by examining the column "computation" in both Tables 1 and 3), that the easiest semantics to compute are given as follows:

- When **positive agent programs with no integrity constraints** are considered, the rational, weak rational, reasonable, weak reasonable, F-preferential, and P-preferential semantics are the easiest to

$\mathcal{IC} = \emptyset$ | $\mathcal{IC}$ *arbitrary*	*consistency*	*computation*	*recognition*	*action reasoning* *possible*	*action reasoning* *certain*
feasible	NP	FNP	P	NP	co-NP
rational	NP | Σ_2^P	FNP//log * | $F\Sigma_2^P$	co-NP	Σ_2^P	co-NP | Π_2^P
reasonable	NP	FNP	P	NP	co-NP
weak rational	NP | Σ_2^P	FNP//log | $FP^{\Sigma_2^P}$ $\cap$ $RP \cdot FP_{\parallel}^{\Sigma_2^P}$ †	co-NP | Π_2^P	Σ_2^P | Σ_3^P	Π_2^P | Π_3^P
weak reasonable	NP	FNP//log	co-NP	Σ_2^P	Π_2^P
F-pref. rational	NP | Σ_2^P	FNP//log | $FP^{\Sigma_2^P}$ $\cap$ $RP \cdot FP_{\parallel}^{\Sigma_2^P}$ †	co-NP | Π_2^P	Σ_2^P | Σ_3^P	Π_2^P | Π_3^P
F-pref. reasonable	NP	FNP//log	co-NP	Σ_2^P	Π_2^P

* … hard for both FNP and $FP_{\parallel}^{NP}$ † … hard for both $F\Sigma_2^P$ and $FP_{\parallel}^{\Sigma_2^P}$

Table 3: Complexity of fixed agent programs with negation

$\mathcal{IC} = \emptyset$ | $\mathcal{IC}$ *arbitrary*	*consistency*	computation	*recognition*	*action reasoning* *possible*	*action reasoning* *certain*
feasible	T 4.1	T 3.8,4.1	P 3.7	T 3.9	T 3.9
rational	T 3.10 | T 4.4	T 3.13 | T 4.4	C 3.12,T 4.2	T 3.14,T 4.5	T 3.14 | T 4.5
reasonable	T 3.16	T 3.16	T 3.15	T 3.17	T 3.17
weak rational	T 3.19 | T 4.11	T 3.20 | T 4.12	T 3.21 | T 4.13	T 3.23 | T 4.14	T 3.23 | T 4.14
weak reasonable	T 3.19,T 4.10	T 3.20,T 4.10	T 3.22	T 3.24	T 3.24
F-preferred rational	C 3.25 | C 4.15	T 3.27 | T 4.18	T 3.26 | T 4.17	T 3.30 | T 4.19	T 3.30 | T 4.19
F-preferred reasonable	C 3.25,C 4.15	T 3.29	T 3.28	T 3.31	T 3.31

Table 4: Location of proofs for Table 3 (C= Corollary, T= Theorem, P= Proposition).

compute, all falling into the same complexity class. The other semantics are harder to compute. Thus, in this case, we have some flexibility in choosing that out of the rational, weak rational, reasonable, weak reasonable, F-preferential, and P-preferential, that best meets the agent's epistemic needs. Note that different agents in IMPACT can use different semantics.

- When **positive agent programs with integrity constraints** are considered, the best semantics, from the point of view of computational complexity, are the rational, reasonable, F-preferential, and P-preferential semantics. Note that unlike the previous case, the weak rational and weak reasonable semantics are harder to compute when integrity constraints are present.
- **When arbitrary agent programs with no integrity constraints** are considered, then the easiest semantics to compute are the feasible set semantics and the reasonable status set semantics. All other semantics are harder to compute.
- **When arbitrary agent programs with integrity constraints** are considered, the same continues to be true.

In general, when considering how to compute a kind of status set, the reasonable status set semantics is generally the easiest to compute, irrespective of whether agent programs are positive or not, and irrespective of whether integrity constraints are present or not. As we have argued earlier on in the paper, reasonable status sets have many nice properties which might make them epistemologically preferable to feasible status sets and rational status sets.

2.3.2 Sources of Complexity

The results show that the complexity of agent programs varies from polynomial up to the third level of the polynomial hierarchy. Observe that in some cases, there are considerable complexity gaps between positive agent programs and agent programs which use negation (e.g., for F-preferred rational status sets).

The reason for this gap are three sources of complexity, which lift the complexity of positive agent programs from P up to Σ_3^P and Π_3^P, respectively (in the cases of F-preferred and weak rational status sets):

1. an (in general) exponential number of candidates for a feasible (resp., weak feasible) status set;
2. a difficult recognition test, which involves groundedness; and
3. an exponential number of preferable candidates, in terms of F-preference or maximal obedience to obligations.

These three sources of complexity act in a way orthogonally to each other; all of them have to be eliminated to gain tractability.

For the canonical semantics of positive agent programs, the rational status set semantics, all computational problems are polynomial. This contrasts with feasible status sets, for which except recognition, all problems are intractable. On the other hand, under the weak status set semantics, the problems (except for action reasoning) are polynomial if no integrity constraints are present; intractability, however, is incurred in all problems as soon as integrity constraints may be used.

It is interesting to observe that for programs with negation, rational status sets are more expensive to compute than reasonable status sets in general, and this is true if no integrity constraints are present, except for consistency checking and cautious action reasoning. A similar observation applies to the F-preferred and weak variants of rational and reasonable status sets in the general case; here, the rational variants are always more complex than the reasonable ones. However, somewhat surprisingly, if no integrity constraints are present, then the complexities of the rational and reasonable variants coincide! This is intuitively explained by the fact that in absence of integrity constraints, the expensive groundedness check for rational status sets can be surpassed in many places, by exploiting the property that in this case, every feasible status set must contain some rational status set.

Another interesting observation is that for programs with negation, the F-preferential and weak variants of rational status sets have the same complexity characteristics, and similar for reasonable status sets. This is explained by the similar optimization components which are present in the semantics, namely minimization of the $\mathbf{F}$-part vs maximization of the set of obligations which are obeyed. These are dual optimization problems, but the underlying optimization principle is the same. A similar complexity behavior is thus not much surprising. However, we note that F-preference and weak rationality are applied to different candidate spaces, namely to all rational status sets versus all A-rational status sets, respectively. This explains that in the case of positive programs, where these candidate spaces have in general different sizes (a singleton set versus an exponential set), the complexity profiles of F-preference and weak rationality are different.

Presence of integrity constraints, even of the simplest form common in practice (e.g., functional dependencies [25] in a database), can have a detrimental effect on (variants of) rational status sets and raises the complexity by one level in the polynomial hierarchy. However, the complexity of reasonable status sets and their variants is immune to integrity constraints except for the weak reasonable status sets on positive programs. Intuitively, this is explained by the fact that the refutation of a candidate for a reasonable status set basically reduces to the computation of the rational status set of a positive agent program, and there integrity constraints do not increase the complexity. In the case of weak reasonable status sets for positive programs, we have an increase since the weakness condition may create an exponential number of candidates if the program is inconsistent.

3 Complexity Results for the Case Without Integrity Constraints

This section contains the first part of the derivation of the complexity results which have been presented in Section 2. The focus in this section is on the base case, in which we have programs without integrity constraints (though cases where results on integrity constraints follow as immediate extensions of the no-integrity-constraint case are also included). As the Table 1 and 3 show, in general the presence of integrity constraints has an effect on the complexity of some problems, while it has not for others. For the latter problems, we discuss this effect in detail in the next section. In this section, as complexity results are discussed, we also develop algorithms for various status set computations.

Before we start with our analysis, we briefly recall the syntax of agent programs. An *agent program* is a finite set of rules

$$A \leftarrow L_1, \ldots, L_n \tag{1}$$

where A is an action status atom and each of $L_1, \ldots, L_n$ is either an action status atom, or a code call atom, each of which may be preceded by a negation sign ($\neg$). A program is *positive*, if it contains no negated action status atoms. *Action status atoms* are of the form $Op(\alpha(t_1, \ldots, t_k))$ where $Op \in \{\mathbf{P}, \mathbf{F}, \mathbf{O}, \mathbf{W}, \mathbf{Do}\}$ is a status modality, α is the name of an action, and $t_1, \ldots, t_k$ are terms (objects or variables) for the action parameters. A *code call atom* represents a call to the software package $\mathcal{S}$, and on instantiating all variables, evaluates to either false or true. Throughout this paper, we shall in programs only encounter code call atoms querying whether a particular tuple t is contained in a table R of a relational database managed by $\mathcal{S}$, i.e., whether the logical fact $R(t)$ is true. Thus, for simplicity, we write $R(t)$ for these code call atoms.

3.1 Positive programs

The most natural question is whether a feasible status set exists for program $\mathcal{P}$ on a given state $\mathcal{O}_\mathcal{S}$. As we have seen, this is not always the case. However, for fixed positive programs, we can always efficiently find a feasible status set (so one exists), and moreover, even a rational status set, measured in the size of the input $\mathcal{O}_\mathcal{S}$. This is possible using the algorithm COMPUTE-P-RSS below, where the program $\mathcal{P}$ and possibly integrity and action constraints are in the background. The algorithm COMPUTE-P-RSS uses the operator $T_{\mathcal{P},\mathcal{O}_\mathcal{S}}$ defined in Part I of this series of papers. We briefly recapitulate its definition:

- The *deontic closure* of a status S, denoted $DCl(S)$, is the closure of S under the rule

  If $\mathbf{O}\alpha \in S$, *then* $\mathbf{P}\alpha \in S$

  where α is any ground action.

- The *action closure* of a status set S, denoted $ACl(S)$, is the closure of S under the rules

  If $\mathbf{O}\alpha \in S$, *then* $\mathbf{Do}\alpha \in S$
  If $\mathbf{Do}\alpha \in S$, *then* $\mathbf{P}\alpha \in S$

  where α is any ground action.

- $App_{\mathcal{P},\mathcal{O}_\mathcal{S}}(S)$ is defined to be the set of all ground action status atoms A such that there exists a rule in P having a ground instance of the form $r : A \leftarrow L_1, \ldots, L_n$ such that

  1. Each positive action status literal L_i is in S and for each negative action status literal $L_i = \neg Op(\alpha)$, α is not in S; and
  2. each positive code call L_i succeeds in $\mathcal{O}_\mathcal{S}$, and
  3. for each negated code call $\neg\chi$, χ does not succeed in $\mathcal{O}_\mathcal{S}$, and
  4. for each positive action status literal $Op(\alpha)$ from $\{A, L_1, \ldots, L_n\}$ such that $Op \in \{\mathbf{P}, \mathbf{O}, \mathbf{Do}\}$, the action α is executable in state $\mathcal{O}_\mathcal{S}$. $\Box$

- For any status set S,

  $$T_{\mathcal{P},\mathcal{O}_\mathcal{S}}(S) = App_{\mathcal{P},\mathcal{O}_\mathcal{S}}(S) \cup DCl(S) \cup ACl(S).$$

Intuitively, $T_{\mathcal{P},\mathcal{O}_\mathcal{S}}(S)$ finds all rules in the agent program $\mathcal{P}$ that are "firable" w.r.t. the current object state $\mathcal{O}_\mathcal{S}$ and w.r.t. the action status atoms in S — it fires such rules to derive the status atoms in the rule heads. This set of status atoms is then closed under the deontic and action closure rules listed above. The reader will easily see that $T_{\mathcal{P},\mathcal{O}_\mathcal{S}}(S)$ can be computed in polynomial time.

Algorithm COMPUTE-PIC-RSS's efficiency can easily be enhanced by interleaving the computation of $lfp(T_{\mathcal{P},\mathcal{O}_\mathcal{S}})$ with the checks in Steps (2) and (3) so as to terminate with failure if a violation of the conditions is detected.

Algorithm COMPUTE-PIC-RSS

Input: agent state $\mathcal{O}_{\mathcal{S}}$ (positive agent program $\mathcal{P}$);

Output: the unique rational status set of $\mathcal{P}$, if it exists; "No", otherwise.

Method

Step 1. Compute $S = \mathit{lfp}(T_{\mathcal{P},\mathcal{O}_{\mathcal{S}}})$;

Step 2. Check whether S satisfies conditions $(S2)$ and $(S4)$ of a feasible status set;

Step 3. If S satisfies $(S2)$ and $(S4)$, then output S; otherwise, output "No". Halt.

The following theorem tells us that when $\mathcal{IC} = \emptyset$, then the problems of checking (decision problem) and finding (search problem) if a positive agent program has a feasible status set is polynomially solvable. Furthermore, as far as the rational status set semantics is concerned, independently of whether $\mathcal{IC}$ is empty or not, it is the case that the consistency and computation problems are polynomial. The reason is that when we consider positive agent programs, the only candidate to be a rational status set is $\mathit{lfp}(T_{\mathcal{P},\mathcal{O}_{\mathcal{S}}})$ which can be computed in polynomial time.

Theorem 3.1 *Let $\mathcal{P}$ be a fixed positive agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_{\mathcal{S}}$, the unique rational status set of $\mathcal{P}$ on $\mathcal{O}_{\mathcal{S}}$ (if it exists) is computed by* COMPUTE-PIC-RSS *in polynomial time. Moreover, if $\mathcal{IC} = \emptyset$, then deciding whether $\mathcal{P}$ has some feasible status set on $\mathcal{O}_{\mathcal{S}}$ as well as computing any such status set, is possible in polynomial time using* COMPUTE-PIC-RSS.

Proof. By [10, Theorem 5.3], a positive $\mathcal{P}$ has over any $\mathcal{O}_{\mathcal{S}}$ a unique rational status set (if a rational status set exists), which is given by $S = \mathit{lfp}(T_{\mathcal{P},\mathcal{O}_{\mathcal{S}}})$ if S is a feasible status set. Since $\mathit{lfp}(T_{\mathcal{P},\mathcal{O}_{\mathcal{S}}})$ satisfies $(S1)$ and $(S3)$ of the definition of feasible status set, algorithm COMPUTE-PIC-RSS correctly computes the unique rational status set of $\mathcal{P}$ on $\mathcal{O}_{\mathcal{S}}$.

By the assumptions that we made in Section 2.1, Step 1 can be done in polynomial time, since a fixed $\mathcal{P}$ amounts to a ground instance which is polynomial in the size of $\mathcal{O}_{\mathcal{S}}$, and we can compute $S = \mathit{lfp}(T_{\mathcal{P},\mathcal{O}_{\mathcal{S}}})$ bottom up by evaluating the sequence $T^i_{\mathcal{P},\mathcal{O}_{\mathcal{S}}}$, $i \geq 0$, until the fixpoint is reached.

Observe that, of course, checking $(S2)$ (action and deontic consistency) –or part of this criterion– in algorithm COMPUTE-PIC-RSS can be done at any time while computing the sequence $T^i_{\mathcal{P},\mathcal{O}_{\mathcal{S}}}$, and the computation can be stopped as soon as an inconsistency is detected.

Step 2, i.e., checking whether S satisfies the conditions $(S2)$ and $(S4)$ is, by our assumptions, possible in polynomial time. Therefore, for fixed $\mathcal{P}$ (and tacitly assumed fixed action and integrity constraints in the background), algorithm COMPUTE-PIC-RSS runs in polynomial time.

If $\mathcal{IC} = \emptyset$, then by [10, Proposition 5.5] $\mathcal{P}$ has a feasible status set on $\mathcal{O}_{\mathcal{S}}$ iff it has a rational status set on $\mathcal{O}_{\mathcal{S}}$. Therefore, deciding the existence of a feasible status set (and computing one) is possible using COMPUTE-PIC-RSS (as any rational status set is feasible) in polynomial time. ∎

The following result is immediately derivable from the preceding one: Given $\mathcal{P}, \mathcal{O}_{\mathcal{S}}$, and a status set S, for checking whether S is rational, we merely need to test whether (i) $S = \mathit{lfp}(T_{\mathcal{P},\mathcal{O}_{\mathcal{S}}})$ and (ii) S satisfies conditions $(S2)$ and $(S4)$ of a feasible status set. The proof of Theorem 3.1 above immediately tells us that these steps are executable in polynomial time.

Algorithm COMPUTE-P-WEAK-RSS

Input: agent state $\mathcal{O}_\mathcal{S}$ (positive agent program $\mathcal{P}$; $\mathcal{IC} = \emptyset$)

Output: a weak rational status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$, if one exists; "No", otherwise.

Method

Step 1. Set $A := \emptyset$, $GA :=$ set of all ground actions, and compute $S := \mathit{lfp}(T_{\mathcal{P},\mathcal{O}_\mathcal{S},A})$;

Step 2. If S is not A-feasible, then output "No" and halt; otherwise, set $A := A(S)$ and $GA := GA \setminus A(S)$;[a]

Step 3. If $GA = \emptyset$, then output S and halt;

Step 4. Choose some ground action $\alpha \in GA$, and set $A' := A \cup \{\alpha\}$;

Step 5. If $S' := \mathit{lfp}(T_{\mathcal{P},\mathcal{O}_\mathcal{S},A'})$ is A'-feasible, then set $A := A(S')$, $GA := GA \setminus A(S')$, and $S := S'$; continue at Step 3.

[a]Recall from Part I of this series of papers that $A(S) = \mathbf{Do}(S) \cup \{\alpha \mid \alpha \notin \mathbf{O}(S)\}$ (see Appendix B).

Corollary 3.2 *Let $\mathcal{P}$ be a fixed positive agent program. Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a status set S, deciding whether S is the rational status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is polynomial.*

As any fixed positive agent program has at most one rational status set, it follows immediately that possible and certain reasoning can be performed in the same time (i.e. polynomial) as it takes to construct such a status set.

Corollary 3.3 *Let $\mathcal{P}$ be a fixed positive agent program. Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a ground action α, deciding whether α is true in some (resp. every) rational status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is polynomial.*

Since for every positive agent program $\mathcal{P}$, the rational status set, the reasonable status set, and their preferred variants coincide, the results for rational status sets in Theorem 3.1 and Corollaries 3.2 and 3.3 extend to these kinds of status sets as well.

3.1.1 Weak rational status sets

In this subsection, we address the problem of computing a weak rational status set for a positive program. As we have mentioned in [10, Section 5.4], for a fixed positive agent program $\mathcal{P}$, computing a weak rational status set on a given agent state $\mathcal{O}_\mathcal{S}$ is possible in polynomial time, provided that no integrity constraints are present. In fact, this is possible by using Algorithm COMPUTE-P-WEAK-RSS shown below.

Before we address the formal correctness of this algorithm, it is useful to consider the associated problem of recognizing a weak rational status set. In general, efficient computability of a solution to a problem does not imply that recognizing a valid solution is also efficiently possible. However, as in the case of rational status set, for a positive program without integrity constraints also recognition of a weak rational status set is polynomial.

Theorem 3.4 *Let $\mathcal{P}$ be a fixed positive agent program, and suppose $\mathcal{IC} = \emptyset$. Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a status set S, deciding whether S is a weak rational status set of $\mathcal{P}$ is polynomial.*

Proof. By [10, Proposition 5.10], every A-feasible status set is $A(S)$-feasible, and thus S must be $A(S)$-feasible if it is a weak rational status set. Since for any set of ground actions A, testing A-feasibility is not harder than testing feasibility, by Proposition 3.7 we obtain that this condition can be tested in polynomial time.

If S is $A(S)$-feasible, then, since $\mathcal{P}$ is positive and $\mathcal{IC} = \emptyset$, by [10, Theorem 5.13] S is a weak rational status set, if and only if $S = \mathit{lfp}(T_{\mathcal{P},\mathcal{O}_\mathcal{S},A})$ and for every ground action $\alpha \notin A(S)$, the status set $S' = \mathit{lfp}(T_{\mathcal{P},\mathcal{O}_\mathcal{S},A'})$ is not A'-feasible, where $A' = A \cup \{\alpha\}$. For each such α, this condition can be checked in polynomial time, and there are only polynomially many such α. Since computing $\mathit{lfp}(T_{\mathcal{P},\mathcal{O}_\mathcal{S},A})$ is polynomial, the overall recognition test is polynomial. ∎

We remark that algorithm COMPUTE-P-WEAK-RSS can be modified to implement the recognition test; we omit the details, however.

The next result states that algorithm COMPUTE-P-WEAK-RSS is correct and polynomial.

Theorem 3.5 *For a positive program $\mathcal{P}$ and an agent state $\mathcal{O}_\mathcal{S}$, algorithm* COMPUTE-P-WEAK-RSS *correctly outputs a weak rational (resp., weak reasonable) status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ (so one exists) if $\mathcal{IC} = \emptyset$. Moreover, for fixed $\mathcal{P}$,* COMPUTE-P-WEAK-RSS *runs in polynomial time.*

Proof. The correctness of the algorithm follows from the arguments used in the proof of Theorem 3.4. Starting from $A = \emptyset$, we can subsequently increase A by a ground action $\alpha \notin A(S)$, until A-feasibility of $S = \mathit{lfp}(T_{\mathcal{P},\mathcal{O}_\mathcal{S},A})$ is no longer possible. The output status set S is then a weak rational status set. ∎

We remark that this simple algorithm can be speeded up by exploiting some further properties. In Step 5 of the algorithm, the computation of S' can be done by least fixpoint iteration starting from S rather than from the empty set (cf. [10, Proposition 5.12]).

As for action reasoning from weak rational status sets, we face for the first time intractable problems in our analysis. The intuitive reason for intractability is that an exponential number of weak rational status sets might exist, all of which must be examined for answering the problem, and there seems no way of efficiently pruning this search space.

Theorem 3.6 *Let $\mathcal{P}$ be a fixed positive agent program, and suppose $\mathcal{IC} = \emptyset$. Let $\mathcal{O}_\mathcal{S}$ be a given agent state and let α be a given ground action. Then, deciding whether $\alpha \in \mathbf{Do}(S)$ holds for (i) every (resp., (i) some) weak rational status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is* co-NP*-complete (resp.,* NP*-complete).*

Proof. Observe that algorithm COMPUTE-P-WEAK-RSS is nondeterministically complete, i.e., every weak rational status set S is produced upon proper choices in Step 4. Therefore, by checking $\mathbf{Do}(\alpha) \notin S$ (resp., $\mathbf{Do}(\alpha) \in S$) before termination, we obtain membership in co-NP (resp., NP).

For the hardness part of (i), we provide a reduction from the complement of problem M3SAT (see Appendix A).

In our reduction, we store the CNF formula $\phi = \bigwedge_i C_i$ in a relational database $\mathcal{D}$. For this purpose, $\mathcal{D}$ is supposed to have two relations POS(V_1, V_2, V_3) and NEG(V_1, V_2, V_3), in which the positive and negative clauses C_i of ϕ are stored, and a relation VAR(V) which contains all variables. For each positive clause C_i, there exists a tuple with the variables of C_i in POS, e.g., for $x_1 \vee x_4 \vee x_2$ the tuple (x_1, x_4, x_2), and likewise for the negative clauses a tuple with the variables in NEG, e.g., for $\neg x_3 \vee \neg x_1 \vee \neg x_2$ the tuple (x_3, x_1, x_2).

The action base $\mathcal{AB}$ contains three actions: $\mathtt{set}_0(X)$, $\mathtt{set}_1(X)$, and α. Here, we assume that every action has empty precondition and empty Add- and Del-set. Define now the program $\mathcal{P}$ as follows.

$$\begin{array}{rcl}
\mathbf{O}(\mathtt{set}_0(X_1)) & \leftarrow & \mathrm{VAR}(X_1) \\
\mathbf{O}(\mathtt{set}_1(X_1)) & \leftarrow & \mathrm{VAR}(X_1) \\
\mathbf{Do}\alpha & \leftarrow & \mathbf{Do}(\mathtt{set}_0(X_1)), \mathbf{Do}(\mathtt{set}_0(X_2)), \mathbf{Do}(\mathtt{set}_0(X_3)), \mathrm{POS}(X_1, X_2, X_3) \\
\mathbf{Do}\alpha & \leftarrow & \mathbf{Do}(\mathtt{set}_1(X_1)), \mathbf{Do}(\mathtt{set}_1(X_2)), \mathbf{Do}(\mathtt{set}_1(X_3)), \mathrm{NEG}(X_1, X_2, X_3)
\end{array}$$

On this program, we impose the following action constraint:

$$AC: \quad \{\mathtt{set}_0(X_1), \mathtt{set}_1(X_1)\} \hookleftarrow \mathrm{VAR}(X_1).$$

We set $\mathcal{AC} = \{AC\}$ and $\mathcal{IC} = \emptyset$. Intuitively, the weak rational status sets correspond to the truth assignment for the variables in X; the maximality of weak rationality and the constraint AC effect that each variable $x_i \in X$ is assigned exaclty one of the values 0 or 1.

For a given database instance D describing a formula ϕ, it is easily seen that every weak rational status set of $\mathcal{P}$ on D contains $\mathbf{Do}\alpha$, if and only if the corresponding M3SAT instance ϕ is a No-instance. Since D is easily constructed from ϕ, part (i) is proved.

For the hardness part of (ii), a similar reduction from M3SAT can be given. We add to the program $\mathcal{P}$ the following clauses:

$$\begin{array}{rcl}
\mathbf{F}\alpha & \leftarrow & \\
\mathbf{Do}(\mathtt{val}(X_1)) & \leftarrow & \mathbf{Do}(\mathtt{set}_0(X_1)), \mathrm{NEXT}(X_1, X_1) \\
\mathbf{Do}(\mathtt{val}(X_1)) & \leftarrow & \mathbf{Do}(\mathtt{set}_1(X_1)), \mathrm{NEXT}(X_1, X_1) \\
\mathbf{Do}(\mathtt{val}(X_1)) & \leftarrow & \mathbf{Do}(\mathtt{set}_0(X_1)), \mathbf{Do}(\mathtt{val}(X_2)), \mathrm{NEXT}(X_1, X_2) \\
\mathbf{Do}(\mathtt{val}(X_1)) & \leftarrow & \mathbf{Do}(\mathtt{set}_1(X_1)), \mathbf{Do}(\mathtt{val}(X_2)), \mathrm{NEXT}(X_1, X_2)
\end{array}$$

The action $\mathtt{val}(X)$ has empty precondition and empty Add- and Del-sets. The database relation $\mathrm{NEXT}(X_1, X_2)$ provides the enumeration of the variables $x_i \in X$, such that database D contains the tuples (x_1, x_2), (x_2, x_3), …,(x_{n-1}, x_n) and (x_n, x_n) for the last variable (which has no successor).

Intuitively, the first clause prohibits the selection of a truth assignment to all variables x_i, if it falsifies the formula ϕ. The other clauses check recursively, starting from the last variable x_n (i.e., $i = n$), whether all variables x_j such that $j \geq i$ have assigned a value. If this is true for $i = 1$, i.e., $\mathbf{Do}(\mathtt{val}(x_1))$ is derived, then all variables have assigned a value.

It holds that $\mathbf{Do}(\mathtt{val}(x_1))$ belongs to some weak rational status set of the augmented program $\mathcal{P}'$ on D if and only if formula ϕ is satisfiable. From this, NP-hardness of (ii) follows. ∎

Before closing this subsection, we remark that tractability of both problems can be asserted, if a total prioritization on the weak rational status sets is used, which technically is derived from a total ordering $\alpha_1 < \alpha_2 < \cdots < \alpha_n$ on the set GA of all ground actions. In this case, a positive agent program $\mathcal{P}$ has a unique weak rational status set S (if one exists). This status set S can be constructed by modifying Step 4 of algorithm COMPUTE-WEAK-RSS as follows:

Step 4′. Let α be the $<$-least action from GA, and set $A' := A \cup \{\alpha\}$.

Thus, in the absence of integrity constraints, the unique weak rational status set can be computed in polynomial time in this case.

3.2 Programs with negation

If we allow unrestricted occurrence of negated status atoms in the rule bodies, then the complexity of evaluating agents programs increases. This is not very surprising, since this way, we can express logical disjunction of positive facts. For example, the rule

$$\mathbf{P}\alpha \leftarrow \neg\mathbf{F}\alpha$$

leads to two rational status sets: $S_1 = \{\mathbf{P}\alpha\}$ and $S_2 = \{\mathbf{F}\alpha\}$. Informally, this clause expresses under rational status set semantics the disjunction $\mathbf{F}\alpha \vee \mathbf{P}\alpha$. Notice that under the reasonable status semantics, the above rule has only a single reasonable status set, namely S_1. However, if we add its contrapositive

$$\mathbf{F}\alpha \leftarrow \neg\mathbf{P}\alpha,$$

then the resulting program has the two reasonable status sets S_1 and S_2. Thus, in the general case, both rational and reasonable status set semantics allow for expressing disjunction, and are for this reason inherently complex. We now analyze the precise complexity of these semantics.

3.2.1 Feasible status sets

We note here that for feasible status sets, the recognition problem is tractable under the assumptions that we made in Subsection 2.1; this can be easily seen as each of the four conditions $(S1)$–$(S4)$ defining feasibility can be polynomially checked.

Proposition 3.7 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a status set S, deciding whether S is a feasible status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is possible in polynomial time.*

However, as the following result shows, the search for feasible status sets is intractable in the general case.

Theorem 3.8 *Let $\mathcal{P}$ be a fixed agent program, and suppose $\mathcal{IC} = \emptyset$. Then, given an agent state $\mathcal{O}_\mathcal{S}$, deciding whether $\mathcal{P}$ has a feasible status set on $\mathcal{O}_\mathcal{S}$ is* NP-*complete, and computing some feasible status set is complete for* FNP.

Proof. By Proposition 3.7, we can guess and check a feasible status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ in polynomial time. Hence, the existence problem is in NP, and the computation problem is in FNP.

To show that the existence problem is NP-hard, we describe a reduction from M3SAT. The reduction is similar to the one in the proof of Theorem 3.6. As there, we suppose that an M3SAT instance ϕ on variables $x_i \in X$ is stored in relations POS (positive clauses) and NEG (negative clauses), and we assume that all variables x_i are stored in VAR. Moreover, we assume that $\mathcal{D}$ has a relation AUX(Var, Val), which contains in the initial database D all tuples $(x_i, 0)$, for all variables x_i.

Now consider the following agent program $\mathcal{P}$:

$$\begin{array}{rcl}
\mathbf{P}\beta & \leftarrow & \\
\mathbf{F}\beta & \leftarrow & \mathbf{F}\alpha(X_1), \mathbf{F}\alpha(X_2), \mathbf{F}\alpha(X_3), \mathrm{POS}(X_1, X_2, X_3) \\
\mathbf{F}\beta & \leftarrow & \mathbf{P}\alpha(X_1), \mathbf{P}\alpha(X_2), \mathbf{P}\alpha(X_3), \mathrm{NEG}(X_1, X_2, X_3) \\
\mathbf{P}\alpha(X_1) & \leftarrow & \neg\mathbf{F}\alpha(X_1), \mathrm{VAR}(X_1)
\end{array}$$

The action base $\mathcal{AB}$ contains two actions α and β, which have both empty preconditions and empty add and delete sets. Thus, these actions do not have any effect on the state of the database. The sets $\mathcal{AC}$ and $\mathcal{IC}$ of action and integrity constraints, respectively, are both assumed to be empty.

Then, it is easy to see that $\mathcal{P}$ possesses a feasible status set over $\mathcal{O}_\mathcal{S}$, if and only if the formula ϕ is satisfiable; the satisfying truth assignments of ϕ correspond naturally (but not 1-1) to the feasible status sets of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$. (Observe that every feasible status set must either contain $\mathbf{P}\alpha(x_i)$ or contain $\mathbf{F}\alpha(x_i)$, for every x_i, but not both; intuitively, $\mathbf{P}\alpha(x_i)$ represents that x_i is true, while $\mathbf{F}\alpha(x_i)$ represents that x_i is false.) Since for a given formula ϕ the database instance D of $\mathcal{D}$ is clearly constructible in polynomial time, it follows that the decision problem is NP-hard. Moreover, by the correspondence between feasible sets status of $\mathcal{P}$ and the satisfying assignments of ϕ, it follows immediately that the feasible status set computation problem is hard for FNP.

Observe that we can replace in the construction the positive atoms $\mathbf{F}\alpha(X_i)$ in the rule with $\mathbf{F}\beta$ in the head by $\neg\mathbf{P}\alpha(X_i)$, and we would get the same feasible status sets; moreover, the last rule could then also be removed, and still a feasible status exists iff ϕ is satisfiable. ∎

This negative result raises the issue of how we can achieve tractability of programs. There are different possibilities.

One possibility is that we identify syntactic constraints under which programs are guaranteed to be tractable. However, as the form of the program in the proof of the previous theorem indicates, rather strict conditions on negation must be imposed, in order to exclude possible inconsistencies. Still, a number of different feasible and rational status sets may exist, due to the inherent logical disjunction. In particular, the reduction in the proof of Theorem 3.8 works for rational status sets as well. In a concurrent piece of work, we have identified a polynomial fragment of agent programs called *regular agent programs* [9]; the implementation of these programs is ongoing.

For action reasoning, we obtain similar intractability results as in the case of weak rational status. This is not surprising, since also here, an exponential number of status sets has to be examined to answer the query.

Theorem 3.9 *Let $\mathcal{P}$ be a fixed agent program. Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a ground action α, deciding whether $\alpha \in \mathbf{Do}(S)$ for (i) every (resp., (ii) some) feasible status set S of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$, is* co-NP-*complete (resp.,* NP-*complete).*

Proof. A guess for a feasible status set S such that $\alpha \notin \mathbf{Do}(S)$ (resp., $\alpha \in \mathbf{Do}(S)$) can be verified in polynomial time (Proposition 3.7).

For the hardness part of (i), observe that the atom $\mathbf{Do}(\beta)$ belongs to every feasible status set of the program $\mathcal{P}$ in the proof of Theorem 3.8, iff $\mathcal{P}$ has no feasible status set. For (ii), we add the rule $\mathbf{Do}\beta \leftarrow$. Then, $\mathbf{Do}(\beta)$ occurs in some feasible status set of the resulting program iff $\mathcal{P}$ has some feasible status set. This proves the result. ∎

3.2.2 Rational status sets

For the consistency problem, we obtain from [10, Proposition 5.5], which states that a rational status set exists just if a feasible status set exists in case $\mathcal{IC} = \emptyset$, and Theorem 3.8 immediately the following result.

Theorem 3.10 *Let $\mathcal{P}$ be a fixed agent program, and suppose $\mathcal{IC} = \emptyset$. Then, given an agent state $\mathcal{O}_\mathcal{S}$, deciding whether $\mathcal{P}$ has a rational status set on $\mathcal{O}_\mathcal{S}$ is* NP-*complete.*

The condition that a feasible status set is grounded requires a minimality check. It turns out that this minimality check is, in general, an expensive operation. In fact, the following holds.

Theorem 3.11 *Let $\mathcal{P}$ be a fixed agent program, and suppose $\mathcal{IC} = \emptyset$. Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a feasible status set S for $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$, deciding whether S is grounded is* co-NP*-complete.*

Proof. In order to refute that S is grounded, we can guess a status set $S' \neq S$ such that $S' \subseteq S$ and verify in polynomial time that S' satisfies the conditions $(S1)$–$(S3)$ of a feasible status set.

To show that the problem is co-NP-hard, we use a variant of the construction in the proof of Theorem 3.8. For the CNF formula ϕ there, we set up the following program $\mathcal{P}$:

$$
\begin{array}{rcl}
\mathbf{P}\beta & \leftarrow & \\
\mathbf{F}\beta & \leftarrow & \neg\mathbf{P}\gamma, \neg\mathbf{P}\alpha(X_1), \neg\mathbf{P}\alpha(X_2), \neg\mathbf{P}\alpha(X_3), \mathrm{POS}(X_1, X_2, X_3) \\
\mathbf{F}\beta & \leftarrow & \neg\mathbf{P}\gamma, \mathbf{P}\alpha(X_1), \mathbf{P}\alpha(X_2), \mathbf{P}\alpha(X_3), \mathrm{NEG}(X_1, X_2, X_3) \\
\mathbf{P}\alpha(X_1) & \leftarrow & \mathbf{P}\gamma, \mathrm{VAR}(X_1)
\end{array}
$$

Here, γ is a new action of the same type as α and β, i.e., it has empty precondition and empty Add- and Del-sets.

It is easily seen that $S = \{\mathbf{P}\beta, \mathbf{P}\gamma\} \cup \{\mathbf{P}\alpha(a_i) \mid i = 1, \ldots, n\}$ is a feasible status set of $\mathcal{P}$. Observe that any feasible status set $S' \neq S$ such that $S' \subseteq S$ must satisfy $\mathbf{P}\gamma \notin S'$. It holds that S is grounded if and only if formula ϕ is not satisfiable. This proves co-NP-hardness.

The reduction even allows to derive another result. In fact, observe that any rational status set of $\mathcal{P}$ is contained in S: if $\mathbf{P}\gamma \in S'$ for a status set S' which satisfies $(S1)$–$(S3)$, then clearly $S' \supseteq S$ holds; otherwise, if $\mathbf{P}\gamma \notin S'$, then $S' \subset S$ must hold. Assume w.l.o.g. that either ϕ is unsatisfiable, or all its satisfying assignments viewed as Boolean vectors are incomparable. Then, S is the unique rational status set of $\mathcal{P}$, iff ϕ is unsatisfiable. This shows that deciding whether an agent program has a unique rational status set is co-NP-hard as well. ∎

The complexity of the recognition problem is an immediate consequence of the previous theorem and Proposition 3.7.

Corollary 3.12 *Let $\mathcal{P}$ be a fixed agent program, and suppose $\mathcal{IC} = \emptyset$. Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a status set S, deciding whether S is a rational status set for $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is* co-NP*-complete.*

In the absence of integrity constraints, the rational status sets coincide with the minimal feasible status sets. Using an NP oracle, we therefore can compute a rational status set using algorithm COMPUTE-RATIONAL-SS. This algorithm correctly outputs a rational status set (so one exists) in polynomial time modulo calls to the oracle. Hence, the problem is in $\mathrm{FP}^{\mathrm{NP}}$. This upper bound can be improved to $\mathrm{FNP}//\log$, since we can nondeterministically compute a rational status set as follows.

1. Compute the smallest size s of a feasible status set S;
2. nondeterministically generate, i.e., guess and check a feasible status set S such that $|S| = s$, and output it.

Step 1 amounts to an NP optimization problem whose output has $O(\log |I|)$ bits: an instance I is given by (fixed) $\mathcal{P}$ and $\mathcal{O}_\mathcal{S}$, and the solutions are the feasible status sets (which are recognizable in polynomial

Algorithm COMPUTE-RATIONAL-SS

Input: agent state $\mathcal{O}_{\mathcal{S}}$ (agent program $\mathcal{P}$, $\mathcal{IC} = \emptyset$);

Output: a rational status set of $\mathcal{P}$, if one exists; "No", otherwise.

Method

Step 1. Set $S := \emptyset$ and GA := set of all ground action status atoms.

Step 2. Check if S is a feasible status set; if true, then output S and halt.

Step 3. If $GA = \emptyset$, then output and halt.

Step 4. Choose some atom $A \in GA$ and query the oracle whether a feasible status set S' exists such that $S \subseteq S' \subseteq S \cup (GA \setminus \{A\})$; If the answer is "no", then $S := S \cup \{A\}$;

Step 5. Set $GA := GA \setminus \{A\}$ and continue at Step 2.

time). The cost of any solution S is its cardinality $|S|$, and deciding whether $s = opt(I) \geq k$ is in NP. Furthermore, s has in binary notation $O(\log |I|)$ many bits. Step 2 is polynomial by Proposition 3.7. Hence, the overall algorithm proves that computing a rational status set is in FNP//log, if $\mathcal{IC} = \emptyset$. We obtain the following result.

Theorem 3.13 *Let $\mathcal{P}$ be a fixed agent program, and suppose $\mathcal{IC} = \emptyset$. Given an agent state $\mathcal{O}_{\mathcal{S}}$, computing any rational status set of $\mathcal{P}$ on $\mathcal{O}_{\mathcal{S}}$ is in* FNP//log *and hard for both* FNP *and* $\mathrm{FP}^{\mathrm{NP}}_{\parallel}$.

Proof. The preceding discussion showed that the problem is in FNP//log. Hardness for FNP follows from the proof of Theorem 3.8 (any rational status set is a feasible status set).

Thus, it remains to show hardness for $\mathrm{FP}^{\mathrm{NP}}_{\parallel}$. We establish this by a reduction of computing a minimal model of a propositional CNF formula ϕ, i.e., find a model M (satisfying truth assignment to the variables), such that no model M' exists with $M' \subset M$, where a model is identified with the set of variables which are true in it. $\mathrm{FP}^{\mathrm{NP}}_{\parallel}$-hardness of this problem, even if all clauses in ϕ have at most three literals, follows easily from the results in [4] (Lemma 4.7).

The reduction is an extension of the one in the proof of Theorem 3.11 (note the observations on rational status sets of the program $\mathcal{P}$ there, and that a rational status set always exists).

We use six further 3-ary relations $C_1, \ldots, C_6$ for storing the clauses which are neither positive nor negative, and add respective rules deriving $\mathbf{F}\beta$. More precisely, if we set C_0 = NEG and C_7 = POS, then the relation C_i stores the clauses $C = L_1 \vee L_2 \vee L_3$ such that the string $p(L_1)p(L_2)p(L_3)$ of the polarities of the literals yields i in binary, where $p(L) = 1$ if L is positive, and $p(L) = 0$, if L is negative; thus, e.g. the clause $x_1 \vee x_5 \vee \neg x_3$ is stored as tuple (x_1, x_5, x_3) in the relation C_6, since $p(x_1)p(x_5)p(\neg x_3) = 110$.

Then, the rational status set of the resulting program $\mathcal{P}'$ on the database D for ϕ correspond 1-1 to the minimal models of ϕ, if ϕ is satisfiable, and the set S from the proof of Theorem 3.11 is the unique rational status set if ϕ is unsatisfiable. Moreover, from any rational status set, the corresponding minimal model $M = \{x_i \mid \mathbf{P}\alpha(x_i) \in S\}$ is easily computed.

Hence, the computation of a minimal model of ϕ reduces to the computation of a rational status set. This implies $\mathrm{FP}^{\mathrm{NP}}_{\parallel}$-hardness, and the theorem is proved. ∎

An improvement of these bounds, in particular completeness for FNP//log, seems to be difficult to achieve. In fact, it can be shown that in case $\mathcal{IC} = \emptyset$ computing a rational status set is equivalent to computing a minimal model of a CNF formula under polynomial-time reductions, which is not known to be complete for FNP//log, cf. [4].

Action reasoning becomes harder in the brave variant if we use rational status sets instead of feasible status sets. The reason is that we have to check groundedness of a status set, which is a source of complexity and adds another level in the polynomial hierarchy. However, for the cautious variant, there is no complexity increase.

Theorem 3.14 *Let $\mathcal{P}$ be a fixed agent program and suppose $\mathcal{IC} = \emptyset$. Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a ground action α, deciding whether $\alpha \in \mathbf{Do}(S)$ holds for (i) every (resp., (ii) some) rational status set S of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is* co-NP*-complete (resp., Σ_2^P-complete).*

Proof. For (i), observe that to disprove $\alpha \in \mathbf{Do}(S)$ for every rational status set S, we can guess a feasible status set S such that $\alpha \notin S$ and verify the guess in polynomial time by Proposition 3.7. Hence, the problem is in co-NP. Hardness follows from the reduction in the proof of Theorem 3.8; there, $\mathbf{Do}(\beta)$ belongs to every rational status set of the constructed program $\mathcal{P}$, if and only if $\mathcal{P}$ has no feasible status set.

The membership part of (ii) is easy: A guess for a rational status set S such that $\alpha \in \mathbf{Do}(S)$ can be verified by Proposition 3.7 and Theorem 3.11 in polynomial time with the help of an NP oracle.

The hardness part is shown by a reduction from evaluating a quantified Boolean formula (QBF) of the form $\forall X \exists Y.\phi$, where ϕ is in M3SAT form (see Appendix A). Telling whether such a formula is false is a well-known Π_2^P-complete problem [12]. The reduction combines the reductions in the proofs of Theorems 3.8 and 3.11 in a suitable way.

We extend the database $\mathcal{D}$ from the proofs of Theorems 3.8 and 3.11, by adding two further relations XVAR and YVAR for storing the variables of X and Y, respectively. Construct a program $\mathcal{P}$, using the actions α, β, and γ from the proof of Theorem 3.11 as follows.

$$
\begin{array}{rcl}
\mathbf{P}\beta & \leftarrow & \\
\mathbf{F}\beta & \leftarrow & \neg\mathbf{P}\gamma, \neg\mathbf{P}\alpha(X_1), \neg\mathbf{P}\alpha(X_2), \neg\mathbf{P}\alpha(X_3), \mathrm{POS}(X_1, X_2, X_3) \\
\mathbf{F}\beta & \leftarrow & \neg\mathbf{P}\gamma, \mathbf{P}\alpha(X_1), \mathbf{P}\alpha(X_2), \mathbf{P}\alpha(X_3), \mathrm{NEG}(X_1, X_2, X_3) \\
\mathbf{P}\alpha(X_1) & \leftarrow & \neg\mathbf{F}\alpha(X_1), \mathrm{XVAR}(X_1) \\
\mathbf{P}\alpha(Y_1) & \leftarrow & \mathbf{P}\gamma, \mathrm{YVAR}(Y_1) \\
\mathbf{Do}\gamma & \leftarrow & \mathbf{P}\gamma
\end{array}
$$

Clearly, every feasible status set S must contain either $\mathbf{P}\alpha(x)$ or $\mathbf{F}\alpha(x)$ (but not both), for every $x \in X$. Moreover, if $\mathbf{P}\gamma \in S$, then $\mathbf{Do}\gamma \in S$ and for all $y \in Y$, we have $\mathbf{P}\alpha(y) \in S$.

Let χ be a choice among the atoms $\mathbf{P}\alpha(x)$ and $\mathbf{F}\alpha(x)$, for all $x \in X$. Then, χ naturally represents a truth assignment to X in which x is *true* if $\mathbf{P}\alpha(x) \in \chi$ and x is *false* if $\mathbf{F}\alpha(x) \in \chi$. Define

$$S_\chi \;=\; \chi \cup \{\mathbf{P}\beta, \mathbf{P}\gamma, \mathbf{Do}\gamma\} \cup \{\mathbf{P}\alpha(y) \mid y \in Y\}.$$

It is easy to see that S_χ is a feasible status set, for every choice χ. We claim that every rational status set S of $\mathcal{P}$ must be contained in some of the S_χ.

To see this, notice that no atoms with status $\mathbf{W}$ or $\mathbf{O}$ can be in S, since there is no possibility to derive such an atom. For the same reason, no atoms $\mathbf{Do}\alpha(v)$, $\mathbf{Do}\beta$, $\mathbf{F}\gamma$ and $\mathbf{F}\alpha(y)$ can be in S, for every $v \in X \cup Y$ and $y \in Y$. Hence, by the observation on $\mathbf{P}\alpha(x_i)$ and $\mathbf{F}\alpha(x_i)$ from above, S must be a subset of some S_χ.

It holds that that S_χ is not grounded, if and only if $\mathbf{P}\gamma$ can be removed from it, such that $S_\chi \setminus \{\mathbf{P}\gamma, \mathbf{Do}\gamma\}$ contains a feasible status set. This happens to be the case if the formula $\exists Y.\phi[X = \chi]$ is true. Thus, it follows that some rational status set of $\mathcal{P}$ contains $\mathbf{Do}\gamma$, if and only if S_χ is a rational status set of $\mathcal{P}$ for some χ, if and only if for some χ the formula $\phi[X = \chi]$ is unsatisfiable, if and only if $\forall X \exists Y.\phi$ is false. Since the database D for $\forall X \exists Y.\phi$ is constructible in polynomial time, this proves (ii) and the theorem. ∎

Of course, for positive agent programs, action reasoning is easier. In fact, in this case it is polynomial for both (i) and (ii) since a rational status set, if it exists, is unique and computable in polynomial time.

3.2.3 Reasonable status sets

Our first result on reasonable status sets is positive: the recognition problem, even in the general setting where we have negation and integrity constraints, is tractable.

Theorem 3.15 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a status set S, deciding whether S is a reasonable status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is possible in polynomial time.*

Proof. Indeed, by our assumptions, the ground instance of $\mathcal{P}$ over the agent state is constructible in polynomial time, and, moreover, the reduct $red^S(\mathcal{P}, \mathcal{O}_\mathcal{S})$ is computable in polynomial time. By Theorem 3.1, the unique rational status set S' of $red^S(\mathcal{P}, \mathcal{O}_\mathcal{S})$ (if S' exists) is computable in polynomial time, and it remains by [10, Theorem 5.3] and the definition of a reasonable status set to check whether $S = S'$. Overall, this is a polynomial-time algorithm. ∎

Computing a reasonable status set, however, is intractable in the general case, even in the absence of integrity constraints. The precise complexity of this and the consistency problem is given in the next result.

Theorem 3.16 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$, deciding whether $\mathcal{P}$ has a reasonable status set on $\mathcal{O}_\mathcal{S}$ is* NP-*complete, and computing some reasonable status set S of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is complete for* FNP. *Hardness holds even if $\mathcal{IC} = \emptyset$.*

Proof. The membership part follows from Theorem 3.15, since a guess for S can be verified in polynomial time.

The hardness part is shown by a slight modification of the reduction in the proof of Theorem 3.8. We add the rule

$$\mathbf{F}\alpha(X_1) \leftarrow \neg\mathbf{P}\alpha(X_1), \mathrm{VAR}(X_1)$$

to the program $\mathcal{P}$ there. Then, the reasonable status sets of the resulting program $\mathcal{P}'$ coincide with the rational status sets of $\mathcal{P}$. This proves the result. (Observe that $\mathcal{P}$ has either no reasonable status set, or a unique such status set; note that $\mathbf{F}\alpha(x_i)$ is not contained in any reasonable status set of $\mathcal{P}$, since there is no possibility for deriving $\mathbf{F}\alpha(x_i)$ by means of the head of a program rule or by deontic closure.) ∎

In the light of this result, it is clear that for non-positive programs without integrity constraints, action reasoning on the reasonable status sets is intractable. However, compared to the rational status sets, the complexity of the brave variant is lower; this is explained by the fact that an expensive groundedness test is dispensable for reasonable status sets, which allows for an efficient recognition.

Theorem 3.17 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a ground action α, deciding whether $\alpha \in \mathbf{Do}(S)$ holds for (i) every (resp., (ii) some) reasonable status set S of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is* co-NP-*complete (resp.,* NP-*complete). Hardness holds even if $\mathcal{IC} = \emptyset$.*

Proof. We can guess a reasonable status set S of $\mathcal{P}$ such that α in $\mathbf{Do}(S)$ (resp., $\alpha \notin \neg\mathbf{Do}(\alpha)$) and verify the guess in polynomial time (Theorem 3.15). This proves the membership part.

Hardness for (i) and (ii) can be easily shown by modifying the reduction in the proof of Theorem 3.8. Add the rule $\mathbf{F}\alpha(X_1) \leftarrow \neg\mathbf{P}\alpha(X_1), \text{VAR}(X_1)$ (cf. proof of Theorem 3.16) and query for (i) about β; for (ii), add a further rule $\mathbf{Do}(\beta) \leftarrow$ and query about β. ∎

3.2.4 Weak status sets

In Subsection 3.1.1, we have already considered the computation of weak rational (resp., weak reasonable) status sets for positive programs. In the presence of negation, the concepts of weak rational status sets and weak reasonable status set do no longer coincide. Also, their complexities are different in general. However, as we shall see, they are the same if no integrity constraints are present.

Recall that compared to rational (resp., reasonable) status sets, we have here to deal with relativized action closure ACl_A, which results in A-feasibility, A-rationality etc. The relativization to A does not affect the complexity.

Proposition 3.18 *Let $\mathcal{P}$ be an agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$, a status set S, and a set of ground actions A, testing A-feasibility of S (resp., A-rationality, A-reasonability), has the same complexity as testing feasibility (resp., rationality, reasonability).*

Since under our assumptions, a weak rational (resp., weak reasonable) status set exists if and only if an A-rational (resp., A-reasonable) status set exists for some A, we easily obtain from Proposition 3.18 and the proofs of Theorems 3.8 and 3.16 the following result.

Theorem 3.19 *Let $\mathcal{P}$ be a fixed agent program, and suppose $\mathcal{IC} = \emptyset$. Then, given an agent state $\mathcal{O}_\mathcal{S}$, deciding whether $\mathcal{P}$ has a weak rational (resp., reasonable) status set on $\mathcal{O}_\mathcal{S}$ is* NP*-complete.*

The computation of any weak rational status set can be accomplished using the algorithm COMPUTE-WEAK-RSS. The steps (1)-(4) can be done in polynomial time with the help of an NP oracle. Therefore, in the absence of integrity constraints computing a weak rational status set is in FP^{NP}. Notice that by [10, Proposition 5.10], which tells that any A-feasible status set is $A(S)$-feasible, the steps 1 and 2 can be combined into computing a status set S which is $A(S)$-feasible and such that $|A(S)|$ is maximal.

For weak reasonable status sets, we can apply an adapted version of COMPUTE-WEAK-RSS, in which "A-feasible" is replaced by "A-reasonable". Notice that the consistency problems for the A-feasible and the A-reasonable status sets have the same complexities.

Thus, for both kinds of status sets, the computation problem is polynomial if an NP oracle may be consulted. We can improve on this upper bound and give an exact characterization of the problem in terms of the complexity class FNP//$\log$, which consists of computation problems with an associated NP optimization problem (see Section 2.2 and [4]).

In our case, this NP optimization problem consists of the computation of the numbers s and s', respectively. It is possible to combine these two steps into a single NP optimization problem Π, such that we can nondeterministically generate, given the optimal value for an instance, a weak rational (resp., reasonable) status set in polynomial time.

Theorem 3.20 *Let $\mathcal{P}$ be a fixed agent program and suppose that $\mathcal{IC} = \emptyset$. Then, computing any weak rational (resp., weak reasonable) status set of $\mathcal{P}$ on a given agent state $\mathcal{O}_\mathcal{S}$ is complete for* FNP//$\log$.

Algorithm COMPUTE-WEAK-RSS

Input: agent state $\mathcal{O}_S$ (agent program $\mathcal{P}, \mathcal{IC} = \emptyset$)

Output: a weak rational status set of $\mathcal{P}$ on $\mathcal{O}_S$, if one exists.

Method

1. Compute the maximum size s of a set A such that $\mathcal{P}$ has an A-feasible status set on $\mathcal{O}_S$;
2. Compute a set A such that $|A| = s$ and some A-feasible status set exists;
3. Compute the smallest size s' of any A-feasible status set S;
4. Compute an A-feasible status set S such that $|S| = s'$, and output S.

Proof. Let GA be the set of all ground actions. Associate with every status set S the tuple $t_S = \langle |A(S)|, |GA| - |S| \rangle$, if S is $A(S)$-feasible, and $t_s = \langle -1, 0 \rangle$ otherwise, and impose on the tuples t_S the usual lexicographic order. Then, the following holds: If S is a status set such that t_S is maximal, then S is a weak rational status set if and only if $t_S \neq \langle -1, 0 \rangle$.

Given a maximal tuple $t_S \neq \langle -1, 0 \rangle$, it is clearly possible to generate a weak rational status set S nondeterministically in polynomial time. Moreover, the tuples t_S can be easily encoded by polynomial size numbers $z(t_S)$, such that $z(t_S) > z(t_{S'})$ iff $t_S > t_{S'}$; e.g., define $z(\langle i, j \rangle) = (|GA|+1)i+j$. Computing the maximum $z(t_S)$ is an NP optimization problem with $O(\log |I|)$ bits: Indeed, the cost $z(t_S)$ of a status set S is computable in polynomial time, and deciding whether $opt(I) = \max_S z(t_s) \geq k$ is in NP. Furthermore, from any $z(t_S)$ the tuple t_S is easily computed. Hence, it follows that computing a weak rational status set is in FNP//log.

It remains to show hardness for this class. For this purpose, we reduce the computation of a X-maximal model of a SAT instance ϕ [3, 4] to this problem (see Appendix A).

The reduction is as follows. Without loss of generality, we assume that ϕ is an M3SAT instance. Indeed, we may split larger clauses by introducing new variables, and exchange positive (resp., negative) literals in clauses by using for each variable x a new variable $\hat{x}$ which is made equivalent to $\neg x$. (All new variables do not belong to the set X.)

The reduction is similar to the one in the proof of Theorem 3.6. We use the action base and database from there, and introduce a further relation XVAR for storing the variables in X. Consider the following program $\mathcal{P}$:

$$
\begin{array}{rcl}
\mathbf{O}(\mathtt{set}_1(X_1)) & \leftarrow & \text{XVAR}(X_1) \\
\mathbf{Do}(\mathtt{set}_0(X_1)) & \leftarrow & \neg\mathbf{Do}(\mathtt{set}_1(X_1)), \text{VAR}(X_1) \\
\mathbf{P}\alpha & \leftarrow & \\
\mathbf{F}\alpha & \leftarrow & \mathbf{Do}(\mathtt{set}_0(X_1)), \mathbf{Do}(\mathtt{set}_0(X_2)), \mathbf{Do}(\mathtt{set}_0(X_3)), \text{POS}(X_1, X_2, X_3) \\
\mathbf{F}\alpha & \leftarrow & \mathbf{Do}(\mathtt{set}_1(X_1)), \mathbf{Do}(\mathtt{set}_1(X_2)), \mathbf{Do}(\mathtt{set}_1(X_3)), \text{NEG}(X_1, X_2, X_3)
\end{array}
$$

and impose on it the action constraint AC:

$$AC: \quad \{\mathtt{set}_0(X_1), \mathtt{set}_1(X_1)\} \hookleftarrow \text{VAR}(X_1).$$

The first rule states that every variable in X should be set to true. The second rule together with the minimality of a A-rational status effects that every variable x_i is set either to true or false, but not both. If the

Algorithm REC-WEAK-RATIONAL

Input: status set S on agent state $\mathcal{O}_{\mathcal{S}}$ (agent program $\mathcal{P}, \mathcal{IC} = \emptyset$)

Output: "Yes", if S is a weak rational status set of $\mathcal{P}$ on $\mathcal{O}_{\mathcal{S}}$, "No" otherwise.

Method

1. Check whether S is $A(S)$-feasible;
2. Check whether there is no $A(S)$-feasible status set S' such that $S' \subset S$;
3. Check whether there is no S' such that S' is $A(S')$-feasible and $A(S) \subset A(S')$.

resulting truth assignment to the variables in X satisfies ϕ, then no deontic inconsistency arises from the last three clauses, and we have an A-rational status set.

It is thus easily seen that the weak rational status sets S of $\mathcal{P}$ on the input database D for an M3SAT instance ϕ correspond 1-1 to the X-maximal models of ϕ. Furthermore, from every such S, the X-part of the corresponding X-maximal model is easily obtained. Since D is efficiently constructed from ϕ in polynomial time, it follows that computing a weak rational status set is hard for FNP//$\log$.

The proof of hardness for computing a weak reasonable status set is similar; we use an additional clause $\mathbf{Do}(\mathtt{set}_1(X_1)) \leftarrow \neg\mathbf{Do}(\mathtt{set}_0(X_1)), \mathrm{VAR}(X_1)$. This proves the result. ∎

As in the case of positive programs, recognition of a weak rational status set S is not harder than computation, even if programs are nonpositive. The recognition problem is solved by algorithm REC-WEAK-RATIONAL. The correctness of this algorithm follows from the definition of weak rational status set and [10, Proposition 5.10]. However, it is not clear how to implement it in polynomial time. The next theorem establishes that such an implementation is unlikely to exist, nor that any polynomial time algorithm for this problem is known.

Theorem 3.21 *Let $\mathcal{P}$ be a fixed agent program and suppose that $\mathcal{IC} = \emptyset$. Then, given an agent state $\mathcal{O}_{\mathcal{S}}$ and a status set S, deciding whether S is a weak rational status set of $\mathcal{P}$ on $\mathcal{O}_{\mathcal{S}}$ is* co-NP*-complete.*

Proof. Algorithm REC-WEAK-RATIONAL can be easily rewritten as a nondeterministic polynomial time algorithm for refuting that S is a weak rational status set. Hardness is immediate from the proof of Theorem 3.11. ∎

A weak reasonable status set can be recognized similar as a weak rational status set. This is accomplished by algorithm REC-WEAK-REASONABLE, whose correctness follows from [10, Proposition 5.10] and the fact that A-reasonable status sets are A-feasible. We obtain the following result.

Theorem 3.22 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_{\mathcal{S}}$ and a status set S, deciding whether S is a weak reasonable status set is* co-NP*-complete. Hardness holds even if $\mathcal{IC} = \emptyset$.*

Proof. Clearly, algorithm REC-WEAK-REASONABLE can be turned into a NP-algorithm for showing that S is not a weak rational status set. Hence, the problem is in co-NP.

The hardness part follows by an easy modification to the proof of Theorem 3.8. Add as in the proof of Theorem 3.16 the rule

$$\mathbf{F}\alpha(X_1) \leftarrow \neg\mathbf{P}\alpha(X_1), \mathrm{VAR}(X_1),$$

Algorithm REC-WEAK-REASONABLE

Input: agent state $\mathcal{O}_S$, status set S (agent program $\mathcal{P}$)

Output: "Yes", if S is a weak reasonable status set of $\mathcal{P}$, "No" otherwise.

Method

1. Check whether S is $A(S)$-reasonable, and output "No" if not;
2. Check whether there is no S' such that S' is $A(S')$-reasonable and $A(S) \subset A(S')$.

and add $\mathbf{O}\beta \leftarrow$. Furthermore, add the atom $\mathbf{Do}\beta$ in the bodies of all rules with head $\mathbf{F}\beta$.

Assume without loss of generality that the truth assignment to X in which every variable x_i is *false* does not satisfy ϕ. Then, $S = \{\mathbf{F}(x_i) \mid x_i \in X\} \cup \{\mathbf{P}\beta, \mathbf{O}\beta\}$ is $A(S)$-reasonable. It is easily seen that S is a weak reasonable status set, if and only if ϕ is not satisfied by any assignment in which some variable x_i is true. (If such an assignment exists, then the obligation $\mathbf{O}\beta$, which is violated in S, can be obeyed, and thus a reasonable status set exists). ∎

As for action reasoning, the complexity of action reasoning is partially affected when we switch from rational (resp., reasonable) status sets to weak versions.

The complexity of brave action reasoning for the weak and the ordinary version of rational status sets is the same if integrity constraints are absent. In both cases, the straightforward Guess-and-Check algorithm yields the same upper bound (Σ_2^P), and the hardness result for brave rational action reasoning has been derived without involving obligations (proof of Theorem 3.14).

For the cautious variant, we find a complexity increase, even if the complexity of the recognition problem has not changed. The reason is that the beneficial monotonicity property of finding just some feasible status set not containing $\mathbf{Do}(\alpha)$, as a proof that $\mathbf{Do}(\alpha)$ does not occur in all rational status sets, can (in any suitable adaptation) no longer be exploited.

Theorem 3.23 *Let $\mathcal{P}$ be a fixed agent program, and suppose $\mathcal{IC} = \emptyset$. Then, given an agent state $\mathcal{O}_S$ and a ground action α, deciding whether $\alpha \in \mathbf{Do}(S)$ holds for (i) every (resp., (ii) some) weak rational status set S of $\mathcal{P}$ on $\mathcal{O}_S$ is Π_2^P-complete (resp., Σ_2^P-complete).*

Proof. The proof of (ii) is in the discussion above.

For the membership part of (i), observe that a weak rational status set S such that $\alpha \notin \mathbf{Do}(S)$ can be guessed and checked by Theorem 3.22 with an NP oracle in polynomial time.

For the hardness part of (i), we adapt the construction in the proof of Theorem 3.20 for a reduction from QBF formulas $\forall X \exists Y.\phi$, where ϕ is in M3SAT form.

We use the action base $\mathcal{AB}$ from there and extend it with another action β of the same type as α. Moreover, we use the relations POS and NEG for storing the clauses of ϕ (cf. proof of Theorem 3.6), and replace VAR by the relations XVAR and YVAR for storing the variables in X and Y, respectively.

Then, we set up the following program:

$$\begin{array}{rcl}
\mathbf{O}(\mathtt{set}_0(X_1)) & \leftarrow & \mathrm{XVAR}(X_1)\\
\mathbf{O}(\mathtt{set}_1(X_1)) & \leftarrow & \mathrm{XVAR}(X_1)\\
\mathbf{Do}(\mathtt{set}_0(Y_1)) & \leftarrow & \neg\mathbf{Do}(\mathtt{set}_1(Y_1)), \mathrm{YVAR}(Y_1)\\
\mathbf{F}\beta & \leftarrow & \mathbf{Do}(\mathtt{set}_0(X_1)), \mathbf{Do}(\mathtt{set}_0(X_2)), \mathbf{Do}(\mathtt{set}_0(X_3)), \mathrm{POS}(X_1, X_2, X_3)\\
\mathbf{F}\beta & \leftarrow & \mathbf{Do}(\mathtt{set}_1(X_1)), \mathbf{Do}(\mathtt{set}_1(X_2)), \mathbf{Do}(\mathtt{set}_1(X_3)), \mathrm{NEG}(X_1, X_2, X_3)\\
\mathbf{O}(\alpha) & \leftarrow & \\
\mathbf{P}(\beta) & \leftarrow & \mathbf{Do}(\alpha)
\end{array}$$

Furthermore, we introduce an action constraint:

$$AC: \quad \{\mathtt{set}_0(X_1), \mathtt{set}_1(X_1)\} \hookleftarrow \mathrm{XVAR}(X_1).$$

In the above program, the agent is informally obliged by the first two clauses to set every variable $x \in X$ to both true and false, which is prohibited by AC. By the maximality of weak rational status set, the agent can safely follow one of the two obligations and assign each variable x_i in X a truth value, which creates an exponential number of possibilities. The subsequent clause, together with the minimality property of an A-rational set, forces she to assign each variable in Y a truth value. The next two clauses check whether the formula ϕ is violated. If so, then $\mathbf{F}\beta$ is derived. In this case, the agent cannot take action α as obliged from the rule $\mathbf{O}(\alpha) \leftarrow$; hence, she must violate this obligation in that case. Thus, if for a choice χ from $\mathbf{O}(\mathtt{set}_0(x_i))$, $\mathbf{O}(\mathtt{set}_1(x_i))$ for all $x_i \in X$ (representing a truth assignment to X), the formula $\phi[X = \chi]$ is unsatisfiable (i.e., $\forall X \exists Y.\phi$ is false), then there exists a weak rational status set S such that $\alpha \notin \mathbf{Do}(S)$. Conversely, if $\alpha \notin \mathbf{Do}(S)$ for such a status set S, then a truth assignment χ to X (given by S) exists such that $\forall Y.\neg\phi[X = \chi]$ is true, i.e., $\forall X \exists Y.\phi$ is false.

Consequently, $\alpha \in \mathbf{Do}(S)$ holds for every weak rational status set of $\mathcal{P}$ on the database D for $\forall X \exists Y.\phi$ if and only if $\forall X \exists Y.\phi$ is true. This proves Π_2^P-hardness of (i) and the result. ∎

For action reasoning with weak reasonable status sets, we obtain similar complexity results.

Theorem 3.24 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a ground action α, deciding whether $\alpha \in \mathbf{Do}(S)$ holds for (i) every (resp., (ii) some) weak reasonable status set S of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is Π_2^P-complete (resp., Σ_2^P-complete). Hardness holds even if $\mathcal{IC} = \emptyset$.*

Proof. A weak reasonable status set S such that $\alpha \notin \mathbf{Do}(S)$ (resp., $\alpha \in \mathbf{Do}(S)$) can be guessed and checked in polynomial time with an NP oracle by Theorem 3.22. This proves membership.

Hardness follows for both problems by a slight extension of the construction in the proof of Theorem 3.23. Add to the program $\mathcal{P}$ there the clause

$$\mathbf{Do}(\mathtt{set}_1(Y_1)) \quad \leftarrow \quad \neg\mathbf{Do}(\mathtt{set}_0(Y_1)), \mathrm{YVAR}(Y_1)$$

Then, the weak reasonable status sets of the resulting program $\mathcal{P}'$ coincide with the weak rational status sets of $\mathcal{P}'$, which coincide with the weak rational status sets of $\mathcal{P}$. This proves the result for (i). For (ii), add the rule $\mathbf{Do}\gamma \leftarrow \neg\mathbf{Do}\alpha$ and query about γ. ∎

3.2.5 Preferred status sets

Intuitively, adding a preference on rational or reasonable status sets does increase the complexity of the semantics. Even if we have checked that a status set S is rational (resp., reasonable), then we still have to verify that there is no other rational (resp., reasonable) status set S' which is preferred over S. This check appears to be expensive, since we have to explore an exponential candidate space of preferred rational (resp., reasonable) status sets S', and the test whether S' is in fact rational is expensive as well.

However, as it turns out, for rational status sets, preference does not lead to an increase in the complexity of action reasoning if $\mathcal{IC} = \emptyset$. On the other hand, preference does increase the complexity of action reasoning for reasonable status set. This is explained by the fact that for rational status sets, an increase in complexity is avoided since for deciding preference, it is sufficient to consider feasible status sets for ruling out a candidate S for a preferred rational set, and feasible status sets have lower complexity. For reasonable status sets, a similar property does not hold, and we enface the situation of being obliged to use reasonable status sets for eliminating a candidate.

In the rest of this paper, we focus on F-preferred status sets. Similar results can be derived for the P-preferred status sets by adapting proofs.

As for the consistency problem, it is clear that under our assumptions a F-preferred rational (resp., reasonable) status set exists just if some rational (resp., reasonable) status exists. Thus, we obtain from Theorems 3.10 and 3.16 the following corollary.

Corollary 3.25 *Let $\mathcal{P}$ be a fixed agent program and suppose $\mathcal{IC} = \emptyset$. Then, given an agent state $\mathcal{O}_\mathcal{S}$, deciding whether $\mathcal{P}$ has some F-preferred rational (resp., reasonable) status set on $\mathcal{O}_\mathcal{S}$ is* NP*-complete.*

For the recognition problem, one might expect a complexity increase if F-preference is applied to rational status sets. However, this is not the case. The reason is that F-preference and groundedness of a rational status set are not orthogonal sources of complexity in the absence of integrity constraints.

Theorem 3.26 *Let $\mathcal{P}$ be a fixed agent program, and suppose $\mathcal{IC} = \emptyset$. Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a status set S, deciding whether S is a F-preferred rational status sets of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is* co-NP*-complete.*

Proof. By Proposition 3.7, one can decide in polynomial time whether S is a feasible status set. Now we exploit the following property: A feasible status set S of $\mathcal{P}$ is not a F-preferred rational status set of $\mathcal{P}$, if and only if there exists a feasible status set S' of $\mathcal{P}$ such that $\mathbf{F}(S') \subset \mathbf{F}(S)$ holds. Therefore, we can refute that S is a F-preferred rational status set by guessing a status set S' and checking in polynomial time whether either S is not a feasible set, or whether S' is feasible and satisfies $\mathbf{F}(S') \subset \mathbf{F}(S)$. Hence, the problem is in co-NP.

Hardness is an immediate consequence of the proof of Theorem 3.11, as the candidate set S defined there satisfies $\mathbf{F}(S) = \emptyset$, and is thus F-preferred, if and only if it is grounded. ∎

The computation of a F-preferred rational status set is possible using a variant of the algorithm COMPUTE-RATIONAL-SS in Section 3.2.2 as follows. After Step 2, compute in a binary search the size s of the smallest possible $\mathbf{F}$-part over all feasible status sets of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$; then, in the remaining steps of the algorithm, constrain the oracle query to existence of a feasible status set S' such that

$$S \subseteq S' \subseteq S \cup (GA \setminus \{A\}) \text{ and } |\mathbf{F}(S')| = s.$$

This is a polynomial algorithm using an NP oracle, and hence the problem is in $\mathrm{FP}^{\mathrm{NP}}$.

A refined analysis unveils that the complexity of this problem, similar to the one of computing a weak rational status set, is captured by the class FNP//log.

Theorem 3.27 *Let $\mathcal{P}$ be a fixed agent program, and suppose $\mathcal{IC} = \emptyset$. Then, given an agent state $\mathcal{O}_\mathcal{S}$, computing an arbitrary F-preferred rational status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is complete for* FNP//log.

Proof. The proof of membership is similar to the one of computing a weak rational status set. Indeed, the F-preferred status sets are those status sets S for which the tuple $t'_S = \langle |\mathbf{F}(S)|, |S|\rangle$ is minimal under lexicographic ordering, where infeasible status sets S have associated the tuple $t'_S = \langle |GA| + 1, |GA|\rangle$, where GA is the set of all ground action status atoms. From a minimal tuple t'_S, a F-preferred rational status set can be constructed by a nondeterministic transducer in polynomial time. Hence, the problem is in FNP//log.

The proof of hardness is by a reduction from the problem X-maximal model in the proof of Theorem 3.20, which is w.l.o.g. in M3SAT form.

We modify the program in the proof of Theorem 3.8 to the following program $\mathcal{P}'$:

$$\begin{array}{rcl}
\mathbf{P}\beta & \leftarrow & \\
\mathbf{F}\beta & \leftarrow & \neg\mathbf{P}\alpha(X_1), \neg\mathbf{P}\alpha(X_2), \neg\mathbf{P}\alpha(X_3), \mathrm{POS}(X_1, X_2, X_3) \\
\mathbf{F}\beta & \leftarrow & \mathbf{P}\alpha(X_1), \mathbf{P}\alpha(X_2), \mathbf{P}\alpha(X_3), \mathrm{NEG}(X_1, X_2, X_3) \\
\mathbf{F}\alpha(X_1) & \leftarrow & \neg\mathbf{P}\alpha(X_1), \mathrm{XVAR}(X_1)
\end{array}$$

Here, XVAR stores the variables in X. The rational status sets of $\mathcal{P}'$ on the database D for ϕ correspond 1-1 to the models M of ϕ such that for the X-part fixed to $X \cap M$, the part of the remaining variables is minimal, i.e., to the models M for which no M' exists such that $M' \cap X = M \cap X$ and $M' \subset M$.

It is not hard to see that for every F-preferred rational status set S of $\mathcal{P}$ on D, the corresponding model M of ϕ is X-maximal. (Observe also that for every X-maximal model ϕ, there exists some F-preferred rational status set S of $\mathcal{P}$ such that the corresponding model M' of ϕ satisfies $M' \cap X = M \cap X$.) Moreover, M is easily constructed from S. It follows that computing an arbitrary F-preferred rational status set is hard for FNP//log. ∎

For F-preferred reasonable status sets, we obtain similar results. However, we may allow the presence of integrity constraints without a change in the complexity.

Theorem 3.28 *Let $\mathcal{P}$ be a fixed agent program. Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a status set S, deciding whether S is a F-preferred reasonable status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is* co-NP-*complete. Hardness holds even if $\mathcal{IC} = \emptyset$.*

Proof. By Theorem 3.15, we can decide in polynomial time whether S is a reasonable status set, and check that there is no reasonable status set S' such that $\mathbf{F}(S') \subset \mathbf{F}(S)$ with the help of an NP oracle.

Hardness is shown by a proper modification of the program $\mathcal{P}$ in the proof of Theorem 3.11. Replace the clause $\mathbf{P}\alpha(X_1) \leftarrow \mathbf{P}\gamma, \mathrm{VAR}(X_1)$ with the clause $\mathbf{F}\alpha(X_1) \leftarrow \mathbf{F}\gamma, \mathrm{VAR}(X_1)$, replace $\neg\mathbf{P}\gamma$ with $\neg\mathbf{F}\gamma$ in the other clauses, and add the following clauses:

$$\begin{array}{rcl}
\mathbf{P}\alpha(X_1) & \leftarrow & \neg\mathbf{F}\alpha(X_1), \mathrm{VAR}(X_1) \\
\mathbf{F}\alpha(X_1) & \leftarrow & \neg\mathbf{P}\alpha(X_1), \mathrm{VAR}(X_1) \\
\mathbf{F}\gamma & \leftarrow & \neg\mathbf{P}\gamma \\
\mathbf{P}\gamma & \leftarrow & \neg\mathbf{F}\gamma
\end{array}$$

Then, the set $S = \{\mathbf{F}\alpha(x_i) \mid x_i \in X\} \cup \{\mathbf{F}\gamma\} \cup \{\mathbf{P}\beta\}$ is a reasonable status set of the new program $\mathcal{P}'$ on the database D. It is the (unique) F-preferred reasonable status set, if and only if the formula ϕ is not satisfiable. Hence, deciding whether S is a F-preferred reasonable status set is co-NP-hard. ∎

A F-preferred reasonable status set can be computed applying an algorithm analogous to the one used for computing a F-preferred rational status set as described above. First, compute the minimum size s of the F-part $\mathbf{F}(S)$ over all reasonable status sets S; then, construct a reasonable status set S such that $|\mathbf{F}(S)| = s$. This matches the solution scheme for problems in FNP//$\log$. Observe that the same algorithm can also be applied in the presence of integrity constraints.

We thus obtain the following result.

Theorem 3.29 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_{\mathcal{S}}$, computing any F-preferred reasonable status set of $\mathcal{P}$ on $\mathcal{O}_{\mathcal{S}}$ is complete for* FNP//$\log$. *Hardness holds even if $\mathcal{IC} = \emptyset$.*

Proof. The membership part is in discussion above. The hardness part is proved by a modification of the reduction in the proof of Theorem 3.27. Add to the program from there the following rules:

$$
\begin{array}{rcl}
\mathbf{P}\alpha(X_1) & \leftarrow & \neg\mathbf{F}\alpha(X_1), \mathrm{XVAR}(X_1) \\
\mathbf{P}\alpha(X_1) & \leftarrow & \neg\mathbf{P}\gamma(X_1), \mathrm{YVAR}(X_1) \\
\mathbf{P}\gamma(X_1) & \leftarrow & \neg\mathbf{P}\alpha(X_1), \mathrm{YVAR}(X_1)
\end{array}
$$

Here, YVAR is a relation which stores the variables which are not in X, and γ is a new action of the same type as α.

It holds that the reasonable status sets of the new program $\mathcal{P}'$ on the database D for ϕ correspond 1-1 to the models of ϕ; moreover, the F-preferred reasonable status sets S correspond 1-1 to the X-maximal models M of ϕ. Since M is easily computed from S, it follows that computing an arbitrary F-preferred reasonable status set is FNP//$\log$-hard. ∎

For action reasoning under F-preferred rational status sets, we obtain similar results as for action reasoning under weak rational status sets. The next result shows that the certainty variant of action reasoning based on F-preferred rational status sets is Π^P_2-complete, while the corresponding possibility variant is (as expected) Σ^P_2-complete.

Theorem 3.30 *Let $\mathcal{P}$ be a fixed program, and suppose that $\mathcal{IC} = \emptyset$. Then, given an agent status $\mathcal{O}_{\mathcal{S}}$ and a ground action α, deciding whether $\alpha \in \mathbf{Do}(S)$ holds for (i) every (resp., (ii) some) F-preferred rational status set S of $\mathcal{P}$ on $\mathcal{O}_{\mathcal{S}}$ is Π^P_2-complete (resp., Σ^P_2-complete).*

Proof. For the membership part, observe that a guess for a F-preferred rational status set S such that $A \notin S$ (resp., $A \in S$), can be verified by checking that F is feasible, F is grounded, and that no feasible status set S' exists such that $\mathbf{F}(S') \subset \mathbf{F}(S)$. By Proposition 3.7, and Theorem 3.11, it follows that these tests can be done in polynomial time with an NP oracle. Hence, the problem is in Π^P_2 (resp., Σ^P_2).

To show Σ^P_2-hardness of (ii), we employ a slight modification of the construction in the proof of case (ii) of Theorem 3.14. Add to the program $\mathcal{P}$ from there the clauses

$$
\begin{array}{rcl}
\mathbf{P}\alpha^*(X_1) & \leftarrow & \mathbf{F}\alpha(X_1), \mathrm{XVAR}(X_1) \\
\mathbf{F}\alpha^*(X_1) & \leftarrow & \mathbf{P}\alpha(X_1), \mathrm{XVAR}(X_1)
\end{array}
$$

where α^* is a new action of the same type as α. The effect of these clauses is to include $\mathbf{P}\alpha^*(x_i)$ in a rational status set, if $\mathbf{F}\alpha(x_i)$ belongs to it, and symmetrically to include $\mathbf{F}\alpha^*(x_i)$, if $\mathbf{P}\alpha(x_i)$ occurs in it.

This way, the choice χ from $\mathbf{F}\alpha(x_i), \mathbf{P}\alpha(x_i)$ is mirrored in a complementary fashion on $\mathbf{F}\alpha^*(x_i), \mathbf{P}\alpha^*(x_i)$ such that the extended candidate set

$$S^*_\chi = S_\chi \cup \{\mathbf{F}\alpha^*(x_) \mid \mathbf{P}\alpha(x_i) \in S_\chi, x_i \in X\} \cup \{\mathbf{P}\alpha^*(x_i) \mid \mathbf{F}\alpha(x_i) \in S_\chi, x_i \in X\}$$

is *not* a F-preferred rational status set, if and only if the formula $\exists Y.\phi[X = \chi]$ is true.

Since any rational status set S' which is F-preferred over S^*_χ must not contain $\mathbf{P}\gamma$ and thus not $\mathbf{Do}\gamma$, it follows that $\mathbf{Do}\gamma$ is contained in some F-preferred rational status set, if and only if the formula $\forall X \exists Y.\phi$ is false. This proves Σ^P_2-hardness of (ii).

For the hardness part of (i), add the clauses $\mathbf{Do}\delta \leftarrow \neg\mathbf{P}\gamma$ and $\mathbf{F}\delta \leftarrow \mathbf{P}\gamma$ to the above program, where δ is another action of the type of α. Then, the set $S^*_\chi \cup \{\mathbf{F}\delta\}$ is a rational status set of the new program, and it contains $\mathbf{Do}\gamma$ but not $\mathbf{Do}\delta$. Every rational status set S' which is F-preferred to $S^*_\chi \cup \{\mathbf{F}\delta\}$ contains $\mathbf{Do}\delta$ and corresponds to some truth assignment to Y which satisfies $\phi[X = \chi]$. It holds that $\mathbf{Do}\delta$ occurs in all F-preferred rational status sets of $\mathcal{P}$ if and only if $\forall X \exists Y.\phi$ is true. This proves Π^P_2-hardness. ∎

As discussed above, F-preference leads for reasonable status sets to a complexity increase, and raises them to the same level as rational status sets. However, as with the other problems on reasonable status sets, this increase is independent of whether integrity constraints are present or not. For rational status sets, this is not the case, and the complexity there is higher in the general case, as we shall see in Section 4.5.

Theorem 3.31 *Let $\mathcal{P}$ be a fixed program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a ground action α, deciding whether $\alpha \in \mathbf{Do}(S)$ holds for (i) every (resp., (ii) some) F-preferred reasonable status set S of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is Π^P_2-complete (resp., Σ^P_2-complete). Hardness holds even if $\mathcal{IC} = \emptyset$.*

Proof. A F-preferred reasonable status set S such that $\alpha \notin \mathbf{Do}(S)$ (resp., $\alpha \notin \mathbf{Do}(S)$) can be guessed and checked in polynomial time with the help of an NP oracle (Theorems 3.15 and 3.28).

For the hardness part, we employ a reduction the QBF problem described in the proof of Theorem 3.14: Decide whether a formula $\Phi = \forall X \exists Y.\, \phi$ where ϕ is a M3SAT instance is true. Without loss of generality, we may assume that $\phi[Y = \emptyset]$ is unsatisfiable.

We extend the reduction in the proof of Theorem 3.8 similar as the one in the proof of Theorem 3.14 for the proof of Theorem 3.30. We construct the following program $\mathcal{P}$, where γ is a fresh action of the same type as β:

$$\begin{array}{rcl}
\mathbf{P}\beta & \leftarrow & \\
\mathbf{F}\beta & \leftarrow & \mathbf{Do}\gamma, \mathbf{F}\alpha(X_1), \mathbf{F}\alpha(X_2), \mathbf{F}\alpha(X_3), \mathrm{POS}(X_1, X_2, X_3) \\
\mathbf{F}\beta & \leftarrow & \mathbf{Do}\gamma, \mathbf{P}\alpha(X_1), \mathbf{P}\alpha(X_2), \mathbf{P}\alpha(X_3), \mathrm{NEG}(X_1, X_2, X_3) \\
\mathbf{P}\alpha(X_1) & \leftarrow & \neg\mathbf{F}\alpha(X_1) \\
\mathbf{F}\alpha(X_1) & \leftarrow & \neg\mathbf{P}\alpha(X_1) \\
\mathbf{P}\alpha^*(X_1) & \leftarrow & \mathbf{F}\alpha(X_1), \mathrm{XVAR}(X_1) \\
\mathbf{F}\alpha^*(X_1) & \leftarrow & \mathbf{P}\alpha(X_1), \mathrm{XVAR}(X_1) \\
\mathbf{Do}(\gamma) & \leftarrow & \neg\mathbf{F}\alpha(Y_1), \mathrm{YVAR}(Y_1)
\end{array}$$

For every choice χ from $\mathbf{F}\alpha(x_i)$,$\mathbf{P}\alpha(x_i)$ for all $x_i \in X$ (representing a truth assignment), the program $\mathcal{P}$ has a reasonable status set S^+_χ which contains $\mathbf{F}\alpha(y_i)$, for all $y_i \in Y$, but does not contain $\mathbf{Do}\gamma$.

Every reasonable status set S' which is F-preferred to S^+_χ must contain the same atoms $\mathbf{F}\alpha(x_i)$ as S^+_χ and encode by a choice from $\mathbf{P}\alpha(y_i)$, $\mathbf{F}\alpha(y_i)$ for all y_i a truth assignment χ' to Y such that not all y_i

are false and $\phi[X = \chi]$ is satisfied by it. Furthermore, by the last clause such a S' must contain $\mathbf{Do}\gamma$. Conversely, if for χ some assignment to Y exists which does not set all y_i to false and satisfies $\phi[X = \chi]$, then a reasonable status set S' exists which is F-preferred to S and contains $\mathbf{Do}\gamma$. It follows that $\mathbf{Do}\gamma$ is contained in every F-preferred reasonable status set of $\mathcal{P}$, if and only if $\forall X(\exists Y \neq \emptyset).\phi$ is true. This proves Π_2^P-hardness of (i).

For (ii), add a further rule $\mathbf{Do}(\delta) \leftarrow \neg\mathbf{Do}(\gamma)$ where δ is a fresh action of the type of β. Then, $\mathbf{Do}(\delta)$ belongs to some F-preferred reasonable status set of $\mathcal{P}$, if and only if $\forall X(\exists Y \neq \emptyset).\phi$ is false. This proves the theorem. ∎

4 Complexity Results for the Case With Integrity Constraints

So far, we have focused in our complexity analysis mainly on agent programs where no integrity constraints on the agent state were present in the background. We say mainly, since for positive programs and reasonable status sets, most results that have been derived in Section 3 do allow for integrity constraints, and fortunately establish tractability for a number of important computation problems.

However, in the presence of negation, we have mostly excluded integrity constraints. The reason is that in some cases, the presence or absence of integrity constraints makes a difference to the intrinsic complexity of a problem, while in other cases, there is no difference. A systematic treatment of this issue is suggestive; therefore, we analyze in this section the effects of integrity constraints on the complexity of agent programs. An overview of the effects and a discussion is given in Section 2.3.

It appears that all problems whose complexities increase in the presence of integrity constraints do so in a very plain setting. Already for a software package $\mathcal{S} = (\mathcal{T}_\mathcal{S}, \mathcal{F}_\mathcal{S})$ which is a simple relational database $\mathcal{D}$ in which tuples may be inserted or deleted from tables, we face these complexity increases if the integrity constraints include *functional dependencies* (FDs for short) on the tables. Notice that FDs are one of the most basic and important type of dependencies in databases [25].[2] All hardness results involving integrity constraints that we derive in this section hold in this setting.

Throughout this section, we adopt in the proofs of hardness results *weakly-concurrent execution* from Part I of this series of papers as the polynomial concurrent execution policy. That is, first all objects which have to be deleted according to the delete sets $Del(\alpha)$ of the taken actions α are removed from the current state, and then the objects which have to be added according to the add sets $Add(\alpha)$ are included in the state.

4.1 Feasible status sets

As shown in the previous section, finding a rational or feasible status set of a positive agent program is polynomial, if no integrity constraints are present. While adding integrity constraints preserves polynomial time computability of rational status sets, it leads to intractability for feasible status sets.

Theorem 4.1 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, deciding whether $\mathcal{P}$ has a feasible status set on a given agent state $\mathcal{O}_\mathcal{S}$ is* NP*-complete, and computing an arbitrary feasible status set is* FNP*-complete. Hardness holds even if $\mathcal{P}$ is positive.*

[2] A functional dependency is a constraint $C : X \rightarrow A$ on a relation r, where A is a column of r and $X = \{X_1, \ldots, X_n\}$ is a subset of columns of r; it holds, if any two tuples in r which agree on the columns in X agree also on A. In our framework, C can be expressed as an integrity constraint e.g. as follows: $\mathtt{in(T1, db : select(r)\&in(T2, db : select(r))\&(T_1.X_1 = T_2.X_1)\& \cdots \&(T_1.X_n = T_2.X_n) \Rightarrow T1.A = T2.A}$.

Proof. The problem is in NP, since a feasible status set S can be guessed and checked in polynomial time, according to our assumptions (cf. Proposition 3.7).

We show the hardness part for the particular restriction by a reduction from the set splitting problem [12]. Given a collection $\mathcal{S} = \{S_1, \ldots, S_m\}$ of nonempty subsets S_i of a finite set U, decide whether there exists a partitioning (or *coloring*) (C_1, C_2) of U such that every $S_i \in \mathcal{S}$, $i = 1, \ldots, m$, meets each of C_1 and C_2 in at least one element, i.e., $|S_i \cap C_1|, |S_i \cap C_2| \geq 1$ holds.

We construct from $\mathcal{S}$ an instance of the feasible status set test as follows. The database $\mathcal{D}$ has four relations: COLL(Set, El), SPLIT$(El, Color)$, A1(Set, El, Tag) and A2(Set, El, Tag). Intuitively, the collection $\mathcal{S}$ is stored in COLL by tuples (i, e) for every $e \in S_i$ and $S_i \in \mathcal{S}$; the table SPLIT is used for placing each element $e \in U$ in C_1 or C_2 (i.e., coloring it), which is indicated by tuples $(e, 1)$ and $(e_2, 2)$; the tables A1 and A2 hold the occurrences of elements in sets, where each set has some tag.

The action base $\mathcal{AB}$ contains actions $\mathtt{assign}(S, X, Y)$ and $\mathtt{trigger}(X, Y)$ as follows:

$$
\begin{array}{ll}
\mathtt{assign}: & Pre(\mathtt{assign}(S,X,Y)) = \mathrm{COLL}(S,X), \\
& Add(\mathtt{assign}(S,X,Y)) = \{\ \mathrm{SPLIT}(X,Y)\}, \\
& Del(\mathtt{assign}(S,X,Y)) = \{\ \mathrm{A1}(S,Z,Y),\ \mathrm{A2}(S,Z,Y)\ \}; \\
& \\
\mathtt{trigger}: & Pre(\mathtt{trigger}(X,Y)) = true, \\
& Add(\mathtt{trigger}(X,Y)) = \{\ \mathrm{A1}(X,Y,0),\ \mathrm{A2}(X,Y,0)\ \}, \\
& Del(\mathtt{trigger}(X,Y)) = \emptyset.
\end{array}
$$

The program $\mathcal{P}$ has the single rule

$$\mathbf{Do}(\mathtt{trigger}(X,Y)) \quad \leftarrow \quad \mathrm{COLL}(X,Y)$$

Let D be the database instance such that COLL contains the collection $\mathcal{S}$, SPLIT is empty, and A1 (resp., A2) holds for each tuple (s, e) in COLL a tuple $(s, e, 1)$ (resp., $(s, e, 2)$). Moreover, suppose that the integrity constraints $\mathcal{IC}$ on $\mathcal{D}$ consist of the following FDs: the FD $El \rightarrow Color$ on SPLIT, and the FD $Set \rightarrow Tag$ on A1 and A2.

Intuitively, the program forces the agent to add for every occurrence of an element in a set $S_i \in \mathcal{S}$, represented by a tuple (i, e) in COLL, a tuple $(i, e, 0)$ to both A1 and A2. This triggers a violation of the FD $Set \rightarrow Tag$ on A1 and A2. This violation must be cured by executing $\mathtt{assign}(i, e_1, 1)$ and $\mathtt{assign}(i, e_2, 2)$ actions for some e_1, e_2 which occur in the set S_i; by the FD $El \rightarrow Color$ on SPLIT, e_1 must be different from e_2. (Notice that, under weakly-current execution, actions $\mathtt{assign}(i, e, 0)$ are useless, since deletions are performed before additions, and this would not cure any violation.)

Hence, it is easy to see that $\mathcal{P}$ has a feasible status set on D, if and only if $\mathcal{S}$ is colorable by some coloring (C_1, C_2). Since a coloring (C_1, C_2) is easily constructed from any feasible status set S, the result follows. ∎

This result is quite negative, since it tells that already for very simple programs and elemantary integrity constraints, computing a feasible set is a hard problem. The reason is that the agent program $\mathcal{P}$ we have constructed in the reduction does not say anything about how and when to use the `assign` action, which does not show up in the program. If we had rules which tell the agent under which conditions a particular `assign` action should be taken or must not be taken, such a situation would not arise. However, since the program is underconstrained in that respect, an exponentiality of possibilities exists which must be explored by the agent.

The previous theorem shows that we benefit from using rational status sets instead of feasible status sets on positive programs in different respects. First, on the semantical side, we have a unique rational status set (if one exists) compared to a possible exponential number of feasible status sets, and second, on the computational side, we can compute the unique rational status set on an agent state in polynomial time, compared to the intractability of computing any feasible status set. Unfortunately, in the presence of negation, like on the semantical side, also on the computational side the appealing properties of rational status sets vanish.

4.2 Rational status sets

The complexity of recognizing a rational status set is not affected by the presence of integrity constraints, since they can be evaluated in polynomial time. The result of Corollary 3.12 thus easily generalizes to this case.

Theorem 4.2 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a status set S, deciding whether S is a rational status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$, is* co-NP*-complete. Hardness holds even if $\mathcal{IC} = \emptyset$.*

On the other hand, computing a rational status set becomes harder if integrity constraints are present, and resides at the second level of the polynomial hierarchy. The reason is that due to the integrity constraints $\mathcal{IC}$, an arbitrary feasible set S may *no longer necessarily contain* a rational status set, and thus picking a feasible status set having smallest size does not necessarily give us a rational status set. In fact, our next result shows that deciding containment of a rational status set is intractable.

Theorem 4.3 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a feasible status set S for $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$, deciding whether S contains some rational status set (resp., S is grounded and thus rational) is* co-NP*-hard, even if $\mathcal{IC}$ contains a single FD.*

Proof. We prove this by a reduction from the M3DNF problem (see Appendix B).

The database $\mathcal{D}$ contains three relations: POS(V_1, V_2, V_3) and NEG(V_1, V_2, V_3) for storing the positive and the negative disjuncts D_i of an M3DNF instance $\phi = \bigvee_i D_i$, respectively, and a relation VAR$(Var, Value, Tag)$, which contains for each pair of a variable $x \in X$ and a value $v \in \{0, 1\}$ precisely one tuple. That is, the FD $Var, Value \to Tag$ is a constraint on VAR.

The initial database D contains the following tuples. For each positive disjunct $D_i = x_{i_1} \wedge x_{i_2} \wedge x_{i_3}$ from ϕ, the tuple $(x_{i_1}, x_{i_2}, x_{i_3})$ is in POS, and for each negative disjunct $D_i = \neg x_{i_1} \wedge \neg x_{i_2} \wedge \neg x_{i_3}$ the tuple $(x_{i_1}, x_{i_2}, x_{i_3})$ is in NEG. Moreover, for each propositional variables $x_i \in X$, the tuples $(x_i, 0, 0)$ and $(x_i, 1, 0)$ are in VAR.

The action base contains the three actions `all`, $\texttt{set}(X, Y)$ and $\texttt{addto\_var}(X, Y, Z)$, which have empty preconditions and the following Add- and Del-sets:

$$\begin{array}{rl}
\texttt{all}: & Add(\texttt{all}) = Del(\texttt{all}) = \emptyset; \\[1em]
\texttt{set}(X,Y): & Add(\texttt{set}(X,Y)) = \emptyset, \\
& Del(\texttt{set}(X,Y)) = \{\text{VAR}(X,Y,0)\}; \\[1em]
\texttt{addto_var}(X,Y,Z): & Add(\texttt{addto_var}(X,Y,Z)) = \{\text{VAR}(X,Y,Z)\}, \\
& Del(X,Y,Z) = \emptyset.
\end{array}$$

The program $\mathcal{P}$ is as follows:

$$\begin{array}{rcl}
\mathbf{Do}(\mathtt{set}(X_1,Y_1)) & \leftarrow & \mathbf{Do}(\mathtt{all}), \mathrm{VAR}(X_1,Y_1,Z_1).\\
\mathbf{Do}(\mathtt{all}) & \leftarrow & \mathbf{Do}(\mathtt{set}(X_1,0)), \mathbf{Do}(\mathtt{set}(X_1,1)), \mathrm{VAR}(X_1,Y_1,Z_1).\\
\mathbf{Do}(\mathtt{all}) & \leftarrow & \neg\mathbf{Do}(\mathtt{set}(X_1,0)), \neg\mathbf{Do}(\mathtt{set}(X_1,1)), \mathrm{VAR}(X_1,Y_1,Z_1).\\
\mathbf{Do}(\mathtt{all}) & \leftarrow & \mathbf{Do}(\mathtt{set}(X,0)), \mathbf{Do}(\mathtt{set}(Y,0)), \mathbf{Do}(\mathtt{set}(Z,0)), \mathrm{POS}(X,Y,Z).\\
\mathbf{Do}(\mathtt{all}) & \leftarrow & \mathbf{Do}(\mathtt{set}(X,1)), \mathbf{Do}(\mathtt{set}(Y,1)), \mathbf{Do}(\mathtt{set}(Z,1)), \mathrm{NEG}(X,Y,Z).\\
\mathbf{Do}(\mathtt{addto_var}(X_1,Y_1,1)) & \leftarrow & \mathrm{VAR}(X_1,Y_1,Z_1).
\end{array}$$

Let S be the smallest status set S which is deontically and action closed such that

$$\mathbf{Do}(S) = \{\mathtt{all}\} \cup \{\mathtt{set}(x_i,v), \mathtt{addto\_var}(x_i,v,1) \mid x_i \in X, v \in \{0,1\}\}.$$

As can be easily checked, S is a feasible status set of $\mathcal{P}$ on the initial database D.

We note that any status set S' such that $S' \subset S$ and the conditions $(S1)$–$(S3)$ of a feasible status set hold must not contain $\mathbf{Do}(\mathtt{all})$, while it must contain exactly one of the atoms $\mathbf{Do}(\mathtt{set}(x_i,0))$, $\mathbf{Do}(\mathtt{set}(x_i,1))$, for every $x_i \in X$. However, any such S' cannot satisfy the FD $Var, Value \to Tag$ on VAR, since either the tuples $(x_i,1,0)$,$(x_i,1,1)$ are in VAR, or the tuples $(x_i,0,0)$, $(x_i,0,1)$ are in VAR, which means that the FD $Var, Value \to Tag$ is violated on VAR.

It holds that S contains some rational status set (resp., that S is grounded), if and only if formula ϕ is a tautology. The result follows. ∎

A straightforward algorithm for computing a rational status set is constructing a feasible status set and checking whether it is grounded. In the light of the previous result, it is unclear how this is possible in polynomial time even if we have an NP oracle. The complexity of computing a rational status set, stated in the next result, is at the second level of the polynomial hierarchy.

Theorem 4.4 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$, deciding whether $\mathcal{P}$ has a rational status set on $\mathcal{O}_\mathcal{S}$ is Σ_2^P-complete, and computing any rational status set is $\mathrm{F}\Sigma_2^P$-complete.*

Proof. The problems are in Σ_2^P and $\mathrm{F}\Sigma_2^P$, respectively, since a rational status set S can be guessed and verified in polynomial time with a call to a NP oracle (cf. Theorem 4.2).

To show that the problems are hard for Σ_2^P and $\mathrm{F}\Sigma_2^P$, respectively, we extend the construction in the proof of Theorem 4.3, such that we encode the problem of computing, given a QBF $\exists Y \forall X.\phi$ where ϕ is in M3DNF (see Appendix A), an assignment χ to the Y-variables such that $\forall X.\phi[Y = \chi]$ is true. This problem is $\mathrm{F}\Sigma_2^P$-complete.

We use an additional relation YVAR for storing the Y-variables, and add the rule

$$\mathbf{Do}(\mathtt{set}(Y_1,1)) \leftarrow \neg\mathbf{Do}(\mathtt{set}(Y_1,0)), \mathrm{YVAR}(Y_1)$$

This rule enforces a choice from $\mathbf{Do}(\mathtt{set}(y_j,0))$ and $\mathbf{Do}(\mathtt{set}(y_j,1))$, for all $y_j \in Y$; each such choice χ (representing a truth assignment to Y), extended by the set S from the proof of Theorem 4.3, generates a candidate S_χ for a rational status set.

It holds that every rational status set of $\mathcal{P}$ on D must be of the form S_χ, for some choice χ; moreover, the rational status sets of $\mathcal{P}$ on D correspond to the sets S_χ such that the formula $\forall X.\phi[Y = \chi]$ is true. Therefore, deciding whether $\mathcal{P}$ has some rational status set on D is Σ_2^P-hard, and computing any rational status set is hard for $\mathrm{F}\Sigma_2^P$. This proves the result. ∎

For action reasoning, we obtain from the preceding theorem easily the following result.

Theorem 4.5 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a ground action α, deciding whether $\alpha \in \mathbf{Do}(S)$ holds for (i) every (resp., (ii) some) rational status set S of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is (i) Π_2^P-complete (resp., (ii) Σ_2^P-complete).*

Proof. Membership is immediate from Theorem 4.2: A guess for a rational status set S such that $\alpha \notin \mathbf{Do}(S)$ (resp., $\alpha \in \mathbf{Do}(S)$) can be verified with a NP oracle in polynomial time.

For the hardness parts, observe that $\mathtt{all} \in \mathbf{Do}(S)$ holds for every rational status set of the program $\mathcal{P}$ in the proof of Theorem 4.4; thus, by querying about `all`, hardness for (i) holds. The hardness part of (ii) follows from Theorem 3.14. ∎

4.3 Reasonable status sets

For reasonable status sets, we find in all cases better computational properties than for rational status sets. This is explained by the fact that the criterion for a reasonable status set is much stronger than the one for a rational status set. Indeed, this criterion is so strong, such that the presence of integrity constraints has no effect on tractability vs intractability issue of recognizing a reasonable status set. In both cases, a reasonable status set can be recognized in polynomial time (Theorem 3.15). Therefore, the same complexity results hold for programs with and without integrity constraints (see Section 3.2.3).

4.4 Weak status sets

The presence of integrity constraints has major effects on the complexity of weak status sets for both positive and arbitrary programs. We thus analyze these two classes of programs in separate sections.

4.4.1 Positive programs

If we impose integrity constraints on the agent state, then recognizing a weak rational status set is no longer polynomial (unless $\mathrm{P} = \mathrm{NP}$). The intuitive reason is that due to the integrity constraints, the maximality of an A-rational status set (and thus weak rationality) is not guaranteed if no further single obligation can be obeyed. If $\mathcal{IC} \neq \emptyset$, then all sets of obligations which have not been respected are relevant, and we end up with an exponential search space in general.

Theorem 4.6 *Let $\mathcal{P}$ be a fixed positive agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a status set S, deciding whether S is a weak rational status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is* $\mathrm{co\text{-}NP}$*-complete.*

Proof. To show that S is not a weak rational status set, we can proceed by [10, Theorem 5.9,Proposition 5.10] as follows. First check whether S is not $A(S)$-rational; if false, i.e., S is $A(S)$-rational, then guess some status set S' such that S' is $A(S')$-rational and $A(S') \supset A(S)$. Since checking A-rationality is polynomial if $\mathcal{P}$ is positive (we need to check whether $S = \mathit{lfp}(T_{\mathcal{P},\mathcal{O}_\mathcal{S},A})$ and S is A-feasible, which is polynomial by Propositions 3.18 and 3.7), membership in $\mathrm{co\text{-}NP}$ follows.

The hardness part is shown by a reduction from problem M3SAT, for which we adapt the one in the proof of Theorem 3.6. As there, the database has relations POS (positive clauses), NEG (negative clauses), and VAR (variables) for storing an M3SAT instance ϕ on variables X. We introduce a further relation AUX(Var, Val), on which we impose the FD $Var \rightarrow Val$.

The initial database D stores ϕ in POS, NEG, and VAR as usual, and AUX contains all tuples $(x_i, 0)$ for $x_i \in X$. Clearly, D satisfies the FD $Var \rightarrow Val$ on AUX.

The action base $\mathcal{AB}$ is modified by setting

$$Add(\mathtt{set}_v(X)) = \{\mathrm{AUX}(Y,1)\} \text{ and } Del(\mathtt{set}_v(X)) = \{\mathrm{AUX}(X,0)\}, \quad v \in \{0,1\}.$$

The modified program $\mathcal{P}$ is as follows.

$$\begin{array}{rcl}
\mathbf{O}(\mathtt{set}_0(X_1)) & \leftarrow & \mathrm{VAR}(X_1) \\
\mathbf{O}(\mathtt{set}_1(X_1)) & \leftarrow & \mathrm{VAR}(X_1) \\
\mathbf{F}\alpha & \leftarrow & \mathbf{Do}(\mathtt{set}_0(X_1)), \mathbf{Do}(\mathtt{set}_0(X_2)), \mathbf{Do}(\mathtt{set}_0(X_3)), \mathrm{POS}(X_1,X_2,X_3) \\
\mathbf{F}\alpha & \leftarrow & \mathbf{Do}(\mathtt{set}_1(X_1)), \mathbf{Do}(\mathtt{set}_1(X_2)), \mathbf{Do}(\mathtt{set}_1(X_3)), \mathrm{NEG}(X_1,X_2,X_3) \\
\mathbf{P}\alpha & \leftarrow &
\end{array}$$

The action constraints $\mathcal{AC}$ contain again the following constraint:

$$AC: \quad \{\mathtt{set}_0(X_1), \mathtt{set}_1(X_1)\} \hookleftarrow \mathrm{VAR}(X_1)$$

It is not hard to see that

$$S = \{\mathbf{O}(\mathtt{set}_0(x_i)),\ \mathbf{O}(\mathtt{set}_1(x_i)),\ \mathbf{P}(\mathtt{set}_0(x_i)),\ \mathbf{P}(\mathtt{set}_1(x_i)) \mid x_i \in X\} \cup \{\mathbf{P}\alpha\}$$

is an $\{\alpha\}$-rational status set of $\mathcal{P}$ on D, and hence by [10, Proposition 5.10] $A(S)$-rational. Moreover, it holds that S is a weak rational status set, if and only if there exists no status set S' such that S' is $A(S')$-rational and $A(S') \supset A(S)$. Observe that any such S' must contain either $\mathbf{Do}(\mathtt{set}_0(x_i))$ or $\mathbf{Do}(\mathtt{set}_1(x_i))$, for every $x_i \in X$, and thus represents a truth assignment to X. Indeed, taking $\mathtt{set}_0(x_i)$ or $\mathtt{set}_1(x_i)$ for any x_i adds the tuples $(x_j, 1)$ to AUX, for all variables $x_j \in X$; for preservation of the FD $Var \to Val$ on AUX, the tuple $(x_j, 0)$ must then be removed from AUX, which requests taking either $\mathtt{set}_0(x_j)$ or $\mathtt{set}_1(x_j)$. On the other hand, for any truth assignment χ to X which satisfies ϕ, a status S' can be obtained such that S' is $A(S')$-rational and $A(S') \supset A(S)$.

Thus, it holds that S is a weak rational status set if and only if ϕ is satisfiable, i.e. a No-instance. Since the database D is easily constructed from ϕ, this proves co-NP-hardness and the result. ∎

As we have seen in Section 3.1.1, a weak rational (resp., reasonable) status set of a fixed positive agent program can be computed in polynomial time using the algorithm COMPUTE-WEAK-RSS. Unfortunately, in the presence of integrity constraints a similar polynomial algorithm is unlikely to exist. This is a consequence of the next result.

Theorem 4.7 *Let $\mathcal{P}$ be a fixed positive agent program (where $\mathcal{IC}$ is arbitrary). Given an agent state $\mathcal{O}_\mathcal{S}$, deciding whether $\mathcal{P}$ has a weak rational status set on $\mathcal{O}_\mathcal{S}$ is* NP*-complete.*

Proof. Under our assumptions, a weak rational status set exists if and only if some A-rational status set exists. By [10, Theorem 5.8], for deciding existence of an A-rational status set we can guess a set A of ground actions, compute $S = lfp(T_{\mathcal{P},\mathcal{O}_\mathcal{S},A})$ and check whether S is A-feasible in polynomial time. Consequently, by Propositions 3.18 and 3.7 the problem is in NP.

NP-hardness is shown by a slight extension to the reduction in the proof of Theorem 4.6. Without loss of generality, the M3SAT formula ϕ there is only satisfiable if a designated variable x_1 is set to true. Thus, if we add the rule $\mathbf{Do}(\mathtt{set}_1(x_1)) \leftarrow$ to the program $\mathcal{P}$, then the resulting program has some weak rational status set if and only if S is not weak rational, if and only if ϕ is satisfiable. ∎

Algorithm COMPUTE-PIC-WEAK-RSS

Input: agent state $\mathcal{O}_{\mathcal{S}}$ (fixed positive agent program $\mathcal{P}$; $\mathcal{IC}$ is arbitrary)

Output: a weak rational status set of $\mathcal{P}$ on $\mathcal{O}_{\mathcal{S}}$, if one exists; "No", otherwise.

Method

Step 1. Set $A_{new} := \emptyset$, GA := set of all ground actions.

Step 2. Query the oracle whether some $A \supseteq A_{new}$ exists such that $S' = \mathit{lfp}(T_{\mathcal{P},\mathcal{O}_{\mathcal{S}},A})$ is $A(S')$-feasible.

Step 3. If the answer is "yes", then let $S := \mathit{lfp}(T_{\mathcal{P},\mathcal{O}_{\mathcal{S}},A_{new}})$ and set $A_{old} := A(S)$, $GA := GA \setminus A_{old}$; otherwise, if $A_{new} = \emptyset$, then output "No" and halt.

Step 4. If $GA = \emptyset$, then output S and halt.

Step 5. choose some $\alpha \in GA$, and set $A_{new} := A_{old} \cup \{\alpha\}$, $GA := GA \setminus \{\alpha\}$; continue at Step 2.

Computing a weak rational status is possible using the algorithm COMPUTE-PIC-WEAK-RSS, which makes use of an oracle. This algorithm computes the last element A_k in a maximal chain $A_0 = \emptyset \subset A_1 \subset \cdots \subset A_k$ of A_i-rational status set, which is a weak rational status set. Its correctness follows from the characterization of weak rational status sets from Part I of this series of papers [10, Section 5.4.1]. The algorithm runs in polynomial time modulo calls to the oracle. The oracle queries are solvable in NP; therefore, computing a weak rational status set is in $\mathrm{FP}^{\mathrm{NP}}$. Observe that in case $\mathcal{IC} = \emptyset$, the NP-oracle can be replaced by a polynomial time algorithm, such that we obtain an overall polynomial algorithm similar to COMPUTE-WEAK-RSS.

Like in other cases, the $\mathrm{FP}^{\mathrm{NP}}$ upper bound for the computation problem can be lowered to $\mathrm{FNP}//\log$ by exploiting nondeterminism.

Theorem 4.8 *Let $\mathcal{P}$ be a fixed positive agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_{\mathcal{S}}$, computing a weak rational status set of $\mathcal{P}$ on $\mathcal{O}_{\mathcal{S}}$ is in* $\mathrm{FNP}//\log$ *and hard for both* FNP *and* $\mathrm{FP}^{\mathrm{NP}}_{\parallel}$.

Proof. A weak rational status set can be computed as follows. First, compute the maximum size $s = |A(S)|$ over all status sets S such that S is $A(S)$-rational; then, generate nondeterministically a status set S which is $A(S)$-rational and such that $|A(S)| = s$, and output this set (so one exists).

The correctness of this algorithm follows from [10, Proposition 5.10]. Since checking whether S is $A(S)$-rational is polynomial if $\mathcal{P}$ is positive (Proposition 3.18 and Corollary 3.2), step 1 of the algorithm amounts to an NP-optimization problem whose output has $O(\log |I|)$ bits. As a consequence, for positive $\mathcal{P}$ computing a weak rational status set is in $\mathrm{FNP}//\log$.

Hardness for FNP follows from the proof of Theorem 4.7: The weak rational status sets of the program from the proof of this theorem correspond to the satisfying assignments of an M3SAT instance, whose computation is easily seen to be FNP-complete.

For the proof of $\mathrm{FP}^{\mathrm{NP}}_{\parallel}$-hardness, we use the fact that given instances $I_1, \ldots, I_n$ of any arbitrary fixed co-NP-complete problem Π, computing the binary string $B = b_1 \cdots b_n$ where $b_i = 1$ if I_i is a Yes-instance of Π and $b_i = 0$ otherwise, is $\mathrm{FP}^{\mathrm{NP}}_{\parallel}$-hard (this is easily seen; cf. also [4, Lemma 4.7]).

We choose for this problem Π the recognition of a weak rational status set S of a fixed positive agent program $\mathcal{P}$, which is co-NP-complete by Theorem 4.6. We assume that $\mathcal{P}$ is the program from the proof of this result, and that S is the status set constructed over the database D constructed for a formula ϕ. We observe that $\mathcal{P}$ has weak rational status set on $\mathcal{P}$, and that S is the unique weak rational status set, iff the formula ϕ is unsatisfiable. Thus, from any arbitrary weak rational status set S' of $\mathcal{P}$ over D, it is immediate whether S is weak rational or not. Consequently, computing weak rational status sets $S_1, \ldots, S_n$ of $\mathcal{P}$ over given databases $D_1, \ldots, D_n$ is $\mathrm{FP}^{\mathrm{NP}}_{\|}$-hard.

It remains to show that the computation of $S_1, \ldots, S_n$ can be reduced to the computation of a single weak rational status set S of a fixed program $\mathcal{P}'$ over a database D'. For thus purpose, we merge the databases D_i into a single database. This is accomplished by tagging each tuple in D_i with i, i.e., we add a new attribute A in each relation, and each tuple from D_i is assigned value i on A. Furthermore, A is added on the left hand side of each functional dependency, an additional argument T for the tag is introduced in each action, and all literals in a rule have the same fresh variable T in the tag position.

The resulting program $\mathcal{P}'$ has some weak rational status set S on the union D' of the tagged D_i's. Moreover, from any such S we can easily extract weak rational status sets $S_1, \ldots, S_n$ of $\mathcal{P}$ on $D_1, \ldots, D_n$ in polynomial time. Since D' is polynomial-time constructible from $D_1, \ldots, D_n$, this proves $\mathrm{FP}^{\mathrm{NP}}_{\|}$-hardness. ∎

For action reasoning, we obtain that integrity constraints cause a complexity increase for the cautious variant. The reason is that, as opposed to the case where $\mathcal{IC} = \emptyset$, it is no longer possible to generate each weak rational status set in nondeterministic polynomial time. For the brave variant, due to monotonicity of positive programs we actually need to consider only A-rational status set for answering the question, which means that maximality of a weak rational status set does not play a role.

Theorem 4.9 *Let $\mathcal{P}$ be a fixed positive agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_S$ and a ground action α, deciding whether $\alpha \in \mathbf{Do}(S)$ holds for (i) every (resp., (ii) some) weak rational status set S of $\mathcal{P}$ on $\mathcal{O}_S$ is Π^P_2-complete (resp., NP-complete).*

Proof. For the membership part of (i), observe that since $\mathcal{P}$ is positive, if S is an A-rational status set, then any A'-rational status set S' such that $A' \supseteq A$ satisfies $S' \supseteq S$. Therefore, for answering the query it suffices to guess a status set S such that S is $A(S)$-rational and $A \in \mathbf{Do}(S)$. By Proposition 3.18 and Corollary 3.2, the guess for S can be verified in polynomial time.

The membership part of (ii) is immediate from Theorem 4.6: A guess for a weak rational status set S such that $A \notin \mathbf{Do}(S)$ can be verified with an NP oracle in polynomial time.

Hardness for (i) follows from Theorem 3.6. The hardness part for (ii) can be shown by a suitable extension of the construction in the proof of Theorem 4.6, such that validity of a QBF $\forall Y \exists X.\, \phi$ is decided, where ϕ is in M3SAT form.

We may assume that no clause of ϕ has all its variables from Y (otherwise, $\forall Y \exists X. \phi$ is trivially false), and that ϕ can only be satisfied if a particular variable $x_1 \in X$ is set to true. We introduce besides POS, NEG, VAR (which stores $X \cup Y$), and AUX (which must satisfy the FD $Var \rightarrow Val$) new relations XVAR and YVAR for storing the variables in X and Y, respectively.

The actions $\mathtt{set}_0(X)$, $\mathtt{set}_1(X)$, and α are modified such that they have empty preconditions and empty Add- and Del-sets. Furthermore, we introduce two new actions $\mathtt{upd}(X, X')$ and $\mathtt{add}(Y)$ as follows:

$$
\begin{array}{llll}
\mathtt{upd}: & Pre(\mathtt{upd}(X,X')) = true, & \mathtt{add}: & Pre(\mathtt{add}(Y)) = true, \\
 & Add(\mathtt{upd}(X,X')) = \{\ \mathrm{AUX}(X',1)\}, & & Add(\mathtt{add}(Y)) = \{\ \mathrm{AUX}(Y,1)\ \}, \\
 & Del(\mathtt{upd}(X,X')) = \{\ \mathrm{AUX}(X,0)\ \}; & & Del(\mathtt{add}(Y)) = \emptyset.
\end{array}
$$

Finally, we add to the program $\mathcal{P}$ in the proof of Theorem 4.6 the following rules (let $v \in \{0,1\}$):

$$\begin{array}{rcl} \mathbf{Do}(\mathtt{add}(Y_1)) & \leftarrow & \mathrm{YVAR}(Y_1) \\ \mathbf{Do}(\mathtt{upd}(Y_1, Y_1)) & \leftarrow & \mathbf{Do}(\mathtt{set}_v(Y_1)), \mathrm{YVAR}(Y_1) \\ \mathbf{Do}(\mathtt{upd}(X_1, X_2)) & \leftarrow & \mathbf{Do}(\mathtt{set}_v(X_1)), \mathrm{XVAR}(X_1), \mathrm{XVAR}(X_2) \end{array}$$

These modifications have the following effect. The first rule adds for each $y_i \in Y$ the tuple $(y_i, 1)$ to AUX and thus causes a violation of the FD $Var \to Val$. This must be cured by executing $\mathtt{upd}(y_i, y_i)$, which requests that y_i is assigned a value (i.e., either $\mathtt{set}_0(y_i)$ or $\mathtt{set}_1(y_i)$ is taken). Assigning a truth value to some variable $x_i \in X$ (i.e., executing $\mathtt{set}_0(x_i)$ or $\mathtt{set}_1(x_i)$) adds a tuple $(x_j, 1)$ to AUX for each $x_j \in X$, whichs causes a violation of the FD $Var \to Tag$ for $x_j \neq x_i$ (observe that $(x_i, 0)$ is removed). Each such violation must be cured by assigning x_j a truth value.

Thus, every weak rational status set of the constructed program on D contains either $\mathbf{Do}(\mathtt{set}_0(y_i))$ or $\mathbf{Do}(\mathtt{set}_1(y_i))$, for each $y_i \in Y$ (but not both), i.e., embodies a choice χ.

On the other hand, for each such choice χ (representing a truth assignment to Y) a weak rational status set exists: If all obligations $\mathbf{O}(\mathtt{set}_0(x_i))$, $\mathbf{O}(\mathtt{set}_1(x_i))$ where $x_i \in X$ are violated, then by the assumption that no clause in ϕ has all its variables from Y, no clause with $\mathbf{F}\gamma$ in the head fires. Hence, we obtain a respective A-rational status set S_χ on D. Since the program is positive, it follows that a weak rational status set $S' \supseteq S_\chi$ exists. It holds that S_χ is weak rational if and only if $\phi[Y = \chi]$ is unsatisfiable. Observe that, by our assumption on x_1, every weak rational S' such that $S' \supset S_\chi$ contains $\mathbf{Do}(\mathtt{set}_1(x_1))$.

It follows that $\mathtt{set}_1(x_1) \in \mathbf{Do}(S)$ holds for every weak rational status set S of the program on D, if and only if the formula $\forall Y \exists X.\phi$ is true. This proves the hardness part for (ii). ∎

4.4.2 Programs with negation

Let us now consider programs with negation. In this case, weak rational and weak reasonable status sets are no longer identical in all cases.

As for weak reasonable status sets, we find that integrity constraints do not add on the complexity. This has already been established for the recognition problem and action reasoning in Theorems 3.22 and 3.24, respectively. It remains to consider the problems of consistency and computation.

Theorem 4.10 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given agent state $\mathcal{O}_\mathcal{S}$, deciding whether $\mathcal{P}$ has a weak reasonable status set on $\mathcal{O}_\mathcal{S}$ is* NP*-complete, and computing any weak reasonable status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is complete for* FNP//log.

Proof. By Theorems 3.19 and 3.20, it remains to prove the membership part. Under the assumptions, a weak reasonable status exists if and only if some A-reasonable status set S exists. Propositions 3.18 and 3.7 imply that deciding $A(S)$-reasonability of S is polynomial. Therefore, a guess for S can be verified in polynomial time. Hence, the consistency problem is in NP.

We can obtain a weak reasonable status set by first computing the maximum s over all $|A(S)|$ such that S is $A(S)$-reasonable, and then generating nondeterministically an $A(S)$-reasonable status set S such that $|A(S)| = s$. Computing s amounts to an NP optimization problem with $O(\log |I|)$ bits; hence, the problem is in FNP//log. ∎

The existence problem of an A-rational status set has the same complexity as the existence problem of a rational status set (Proposition 3.18). Since a weak rational status set exists if and only if an A-rational status set exists for some A, we obtain from the proof of Theorem 4.4 (which does not involve obligations) the following result.

Theorem 4.11 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$, deciding whether $\mathcal{P}$ has a weak rational status set on $\mathcal{O}_\mathcal{S}$ is Σ_2^P-complete.*

For the computation of a weak rational status set, we can use a modified version of the algorithm COMPUTE-WEAK-RSS in Section 3.2.4: Replace in it "A-feasible" globally through "A-rational". This increases the complexity, as we have to replace the NP oracle by a Σ_2^P oracle. Overall, we now have a polynomial time computation which uses a Σ_2^P oracle; consequently, the problem belongs to $\mathrm{FP}^{\Sigma_2^P}$. This can be complemented by a probabilistic upper bound.

Theorem 4.12 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, computing any weak rational status set of $\mathcal{P}$ on a given agent state $\mathcal{O}_\mathcal{S}$ is in $\mathrm{FP}^{\Sigma_2^P} \cap \mathrm{RP}\cdot\mathrm{FP}_{\parallel}^{\Sigma_2^P}$ and hard for both $\mathrm{F}\Sigma_2^P$ and $\mathrm{FP}_{\parallel}^{\Sigma_2^P}$.*

Proof. Membership in $\mathrm{FP}^{\Sigma_2^P}$ was discussed above. Membership in $\mathrm{RP}\cdot\mathrm{FP}_{\parallel}^{\Sigma_2^P}$ can be established using results from [4]. In fact, the computation of a weak rational status set in the general case can be easily expressed as a maximization problem (MAXP) as defined in [4], such that the instance-solution relation is co-NP-decidable; for such problems, $\mathrm{RP}\cdot\mathrm{FP}_{\parallel}^{\Sigma_2^P}$ is an upper bound [4].

Hardness for $\mathrm{F}\Sigma_2^P$ is immediate from the proof of Theorem 4.4, since the operator $\mathbf{O}$ does not occur in the program constructed. Hardness for $\mathrm{FP}_{\parallel}^{\Sigma_2^P}$ can be established as follows. Let Π be any Σ_2^P-complete problem. Then, given instances $I_1, \ldots, I_n$ of Π, computing the binary string $B = b_1 \cdots b_n$ where $b_i = 1$ if I_i is a Yest-instance and $b_i = 0$ otherwise, is easily seen to be hard for $\mathrm{FP}_{\parallel}^{\Sigma_2^P}$.

From the proof of Theorem 4.4, we know that deciding whether a fixed agent program $\mathcal{P}$ in which the operator $\mathbf{O}$ does not occur, has a rational status set on a given database D is Σ_2^P complete. Thus, for given databases $D_1, \ldots, D_n$, computing the string B is $\mathrm{FP}_{\parallel}^{\Sigma_2^P}$-hard.

The different instances can be combined into a single instance of a new fixed program as follows. Take a fresh action α, which does not occur in $\mathcal{P}$ and has empty precondition and Add- and Del-sets. Add the atom $\mathbf{Do}\alpha$ in the body of each rule in $\mathcal{P}$, and add the rule $\mathbf{O}\alpha \leftarrow$. Then the resulting program $\mathcal{P}_0$ has some weak rational status set S on each D_i, and for any such S it holds that $\alpha \in \mathbf{Do}(S)$ iff $\mathcal{P}_0$ has a rational status set on D_i.

The databases D_i can be merged into a single database D' for a new fixed program $\mathcal{P}'$, in the same way as described in the proof of Theorem 4.8, by tagging the databases D_i with i and taking their union. This program $\mathcal{P}'$ has some weak rational status set S on D'; moreover, for every such S, it holds that $\alpha(i) \in \mathbf{Do}(S)$ iff $\mathcal{P}$ has a rational status set on D_i; thus, from any weak rational status set S the binary string B is easily computed.

Since the database D' is polynomial-time constructible from $D_1, \ldots, D_n$, it follows that computing a weak rational status set is hard for $\mathrm{FP}_{\parallel}^{\Sigma_2^P}$. ∎

We next consider the recognition problem. Here, the complexity increases if integrity constraints are allowed; the benign property that an A-feasible status set is A-rational, if no smaller A-feasible status set exists is no longer valid.

Theorem 4.13 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a status set S, deciding whether S is a weak rational status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is Π_2^P-complete.*

Proof. For the membership part, consider the following algorithm for disproving that S is a weak rational status set. First, check whether S is not an $A(S)$-rational status set. If S is found $A(S)$-rational, then guess $A' \supset A(S)$ and S' and check whether S' is A'-rational. Since checking A-rationality of S is by Proposition 3.18 and Theorem 4.2 in co-NP, this is a nondeterministic polynomial algorithm using an NP-oracle. Hence, the problem is in Π_2^P.

For the hardness part, we adapt the construction in the proof of Theorem 4.3 for QBF formulas $\exists Y \forall X.\phi$, by adding the $\exists Y$ quantifier block.

We use the database $\mathcal{D}$, the actions base $\mathcal{AB}$, and the integrity constraints as there, but add to $\mathcal{D}$ another relation YVAR for storing the Y-variables (the X-variables are in VAR) and introduce another action α, which has empty precondition and empty Add- and Del-sets.

We add the following clauses in the program:

$$
\begin{array}{rcl}
\mathbf{O}(\alpha) & \leftarrow & \\
\mathbf{O}(\mathtt{set}(Y_1, 0)) & \leftarrow & \mathrm{YVAR}(Y_1) \\
\mathbf{O}(\mathtt{set}(Y_1, 1)) & \leftarrow & \mathrm{YVAR}(Y_1) \\
\mathbf{Do}(\mathtt{set}(Y_1, 0)), & \leftarrow & \mathbf{Do}(\alpha), \neg\mathbf{Do}(\mathtt{set}(Y_1, 1)), \mathrm{YVAR}(Y_1) \\
\mathbf{Do}(\alpha) & \leftarrow & \mathbf{Do}(\mathtt{set}(Y_1, 0)), \mathrm{YVAR}(Y_1) \\
\mathbf{Do}(\alpha) & \leftarrow & \mathbf{Do}(\mathtt{set}(Y_1, 1)), \mathrm{YVAR}(Y_1)
\end{array}
$$

Let the resulting program be $\mathcal{P}'$, and set up the action constraint:

$$AC: \quad \{\mathtt{set}(Y_1, 0), \mathtt{set}(Y_1, 1)\} \hookleftarrow \mathrm{YVAR}(Y_1).$$

The additional rules state that the agent is obliged to execute α and to set every variable $y_i \in Y$ to false (0) and true (1), which is however prohibited by AC. Moreover, each y_i must have assigned a value if α is executed, and if some variable receives a value, then α is executed. Consequently, if α is executed, then every y_i gets precisely one value, and if α is not executed, then no y_i gets a value.

Let S_0 be the status sets defined by

$$S_0 = S \cup \{\mathbf{O}\alpha, \mathbf{P}\alpha\} \cup \{\mathbf{O}(\mathtt{set}(y_i, v)), \mathbf{P}(\mathtt{set}(y_i, v)) \mid y_i \in Y, v \in \{0, 1\}\},$$

where S is the status set from the proof of Theorem 4.3. Then, S_0 is an $A(S_0)$-rational status set, in which all the obligations from the newly added rules are violated.

It holds that S_0 is the (unique) weak rational status set of $\mathcal{P}'$ iff $\forall Y \exists X. \neg\phi$ is true.

$(\Rightarrow)$ Suppose S_0 is weak rational. Then, for any choice χ from $\mathbf{Do}(\mathtt{set}(y_i, 0))$, $\mathbf{Do}(\mathtt{set}(y_i, 1))$, for all $y_i \in Y$ (representing a truth assignment to Y), it is impossible to find an A-rational status set such that the obligations followed in A include χ. In particular, the status set

$$S_\chi = S_0 \cup \chi \cup \{\mathbf{Do}\alpha\}$$

is not weak rational. As easily checked, S_χ is $A(S_\chi)$-feasible; hence, some $S' \subset S$ must exist which satisfies the conditions $(S1)$–$(S3)$ of $A(S_\chi)$-feasibility. Consequently, $\forall X. \phi[Y = \chi]$ is false. It follows that $\forall Y \exists X. \neg\phi$ is true.

$(\Leftarrow)$ Suppose $\forall Y \exists X. \neg\phi$ is true. Consider any weak rational status set S of $\mathcal{P}'$. Then, either (i) $A(S)$ defines a choice χ from $\mathbf{Do}(\mathtt{set}(y_i, 0))$, $\mathbf{Do}(\mathtt{set}(y_i, 1))$, for all $y_i \in Y$, and $\alpha \in \mathbf{Do}(S)$, or (ii) $A(S) = A(S_0)$.

Assume that (i) is true and consider the following two cases:

(1) $\mathbf{Do}(\texttt{all}) \notin S$. Then, exactly one of the actions $\texttt{set}(x_i, 0)$, $\texttt{set}(x_i, 1)$ must be in $\mathbf{Do}(S)$, for every $x_i \in X$. But then, executing $\mathbf{Do}(S)$ violates the integrity constraint $Var, Value \to Tag$ on VAR. This means that S is not a $A(S)$-rational status set, which contradicts that S is weak rational.

(2) $\mathbf{Do}(\texttt{all}) \in S$. Since by assumption $\forall X.\, \phi[Y = \chi]$ is false, there exists some $S' \subset S$ which satisfies the conditions $(S1)$–$(S3)$ of $A(S)$-feasibility. Again, this means that S is not $A(S)$-rational and thus contradicts weak rationality of S.

Hence, case (i) is impossible, and thus case (ii) must apply to S. Consequently, S_0 is a weak rational status set. It can be easily seen that $S = S_0$ must hold. This proves the result. ∎

The last result that we turn to in this subsection is action reasoning under weak rational status sets. Here we face the full complexity of all conditions that we have imposed on acceptable status sets.

Theorem 4.14 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Let $\mathcal{O}_\mathcal{S}$ be a given agent state and let α be a given ground action. Then, deciding whether $\alpha \in \mathbf{Do}(S)$ holds for (i) every (resp., (ii) some) weak rational status set S of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is Π^P_3-complete (resp., Σ^P_3-complete).*

Proof. The membership part is routine: A guess for a weak rational status set S such that $\alpha \notin \mathbf{Do}(S)$ (resp., $\alpha \in \mathbf{Do}(S)$) can be verified with a Σ^P_2 oracle in polynomial time (Theorem 4.13).

For the hardness part, we extend the construction in the proof of Theorem 4.13 to QBF formulas $\forall Z \exists Y \forall X.\phi$, by adding another quantifier block.

For that, we introduce a new relation ZVAR for storing the variables in Z, and add the following clauses to the program $\mathcal{P}'$ from the proof of Theorem 4.13:

$$\begin{array}{rcl}\mathbf{O}(\texttt{set}(Z_1, 0)) & \leftarrow & \text{ZVAR}(Z_1) \\ \mathbf{O}(\texttt{set}(Z_1, 1)) & \leftarrow & \text{ZVAR}(Z_1) \\ \mathbf{Do}(\texttt{set}(Z_1, 0)), & \leftarrow & \neg\mathbf{Do}(\texttt{set}(Z_1, 1)), \text{ZVAR}(Z_1)\end{array}$$

Denote the resulting program by $\mathcal{P}''$. Moreover, we add another action constraint

$$AC': \quad \{\texttt{set}(Z_1, 0), \texttt{set}(Z_1, 1)\} \hookleftarrow \text{ZVAR}(Z_1).$$

Similar as the rules for the variables in Y, the new rules and AC' force the agent to make a choice χ from $\mathbf{Do}(\texttt{set}(z_i, 0), \mathbf{Do}(\texttt{set}(z_i, 1))$, for all $z_i \in Z$ (representing a truth assignment to Z) in every weak rational status set. Upon such a choice, the program $\mathcal{P}''$ behaves like the program $\mathcal{P}'$. Thus, for any such choice χ, a weak rational status set S including χ contains $\mathbf{Do}\alpha$ if and only if $\exists Y \forall X \phi[Z = \chi]$ is true.

It follows that $\mathbf{Do}\alpha$ belongs to every weak rational status set of $\mathcal{P}''$ if and only if $\forall Z \exists Y \forall X.\phi$ is true. This proves Π^P_3-hardness of (i).

For (ii), we add the rule $\mathbf{Do}(\beta) \leftarrow \neg\mathbf{Do}(\alpha)$ in the program, where β is a fresh action of the type of α. Let $\mathcal{P}^*$ be the resulting program.

It holds that $\mathcal{P}^*$ has some weak rational status set containing $\mathbf{Do}\beta$ if and only if $\mathcal{P}''$ has some weak rational status set not containing $\mathbf{Do}\alpha$. This implies Σ^P_3-hardness of (ii). ∎

4.5 Preferred status sets

Imposing integrity constraint on the agent state acts differently on rational versus reasonable status sets. It appears that for rational status sets, we have a complexity increase, while for reasonable status sets, the complexity remains unchanged (see Section 3.2.5).

As for the consistency problem, under the assumptions a F-preferred rational (resp., reasonable) status set exists if and only if a rational (resp., reasonable) status set exists. Thus, from Theorems 4.4 and 3.16, we obtain the following corollary.

Corollary 4.15 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$, deciding whether $\mathcal{P}$ has some F-preferred rational (resp., reasonable) status set on $\mathcal{O}_\mathcal{S}$ is Σ_2^P-complete (resp.,* NP*-complete).*

In the presence of integrity constraints, the recognition problem for F-preferred rational status sets migrates to the next level of the polynomial hierarchy. For the proof of this result, we use the following convenient lemma.

Lemma 4.16 (see Appendix C) *Let $\Phi' = \forall Z \exists Y' \forall X' \phi'$ be a QBF such that ϕ' is in DNF, where Z may be void. Then, a formula $\Phi = \forall Z \exists Y \forall X.\, \phi$, where ϕ is in M3DNF is constructible in polynomial time, such that*

(1) *the formula $\forall ZX.\, \phi[Y = \emptyset]$ is true;*

(2) *Φ' is equivalent to $\forall Z(\exists Y \neq \emptyset)\forall X.\, \phi$.*

Theorem 4.17 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$ and a status set S, deciding whether S is a F-preferred rational status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ is Π_2^P-complete.*

Proof. Checking whether S is a rational status set can be done with a call to an NP oracle (Theorem 4.2), and a rational status set S' such that $\mathbf{F}(S') \subset \mathbf{F}(S)$ can be guessed an checked in polynomial time with an NP oracle; hence, showing that S is not a F-preferred rational status set is in Σ_2^P.

The proof of hardness is an extension to the proof of Theorem 4.3. We encode evaluating a QBF $\exists Y \forall X.\phi$ where ϕ is in M3DNF form, by adding an alternate block of quantifiers $\exists Y$ in the construction.

For convenience, we may start from the formula $\exists Y \forall X.\phi$ as in Lemma 4.16, and encode the failure of condition (2) of it; this is a Π_2^P-hard problem.

We use the database relations POS and NEG from the proof of Theorem 4.3 for storing the positive and negative disjuncts of ϕ, respectively. The relation VAR however, is extended by an additional tag column to the format VAR$(Var, Value, Tag, Tag1)$. The additional tag $Tag1$ indicates whether a variable is from X (value 0) or from Y (value 1).

We impose two FDs on VAR: $Var, Value \to Tag$ and $Var \to Tag1$.

The action base $\mathcal{AB}$ is the same, with the only difference that $\mathtt{addto\_var}$ has a fourth parameter W and that each occurrence of VAR(X, Y, Z) has to be replaced by VAR(X, Y, Z, W).

We modify and extend the program $\mathcal{P}$ to a program $\mathcal{P}'$ as follows. First, each occurrence of an atom VAR(X, Y, Z) is replaced by VAR$(X, Y, Z, 0)$, and $\mathtt{addto\_var}(X, Y, 1)$ is replaced by $\mathtt{addto\_var}(X, Y, 1, 0)$. We then add the rules

$$\begin{array}{lrcl} (1) & \mathbf{F}(\mathtt{set}(X_1, 1)) & \leftarrow & \mathbf{Do}(\mathtt{set}(X_1, 0)), \mathrm{VAR}(X_1, Y, Z, 1) \\ (2) & \mathbf{Do}(\mathtt{set}(X_1, 1)) & \leftarrow & \neg\mathbf{Do}(\mathtt{set}(X_1, 0)), \mathrm{VAR}(X_1, Y_1, Z_1, 1) \end{array}$$

For each choice χ from $\mathbf{Do}(\mathtt{set}(y_i, 0))$, $\mathbf{Do}(\mathtt{set}(y_i, 1))$, for all $y_i \in Y$ (representing a truth assignment to Y), we obtain a candidate S_χ for a F-preferred rational status set which is a feasible status set of $\mathcal{P}'$. This candidate is the deontic and action closure of the set

$$S \cup \chi \cup \{\mathbf{F}(\mathtt{set}(y_i, 1)) \mid \mathtt{set}(y_i, 0) \in \mathbf{Do}(\chi)\},$$

where S is the feasible status set from the proof of Theorem 4.3 (adapted for addto_var).

For the choice $\chi = \chi_0$ which sets all y_i to *false*, S_χ has a maximal F-part over all χ; this is also the maximal F-part possible for any rational status set of $\mathcal{P}'$. Moreover, by Lemma 4.16 and the construction of $\mathcal{P}'$, this S_{χ_0} is a rational status set.

We claim that S_{χ_0} is F-preferred if and only if $\exists Y \neq \emptyset \forall X.\, \phi$ is false. This proves the result.

($\Rightarrow$) Suppose S_{χ_0} is F-preferred. Then, no rational status set S' exists such that $\mathbf{F}(S') \subset \mathbf{F}(S)$. In particular, for every $\chi \neq \chi_0$ the set S_χ is not a rational status set. As in the proof of Theorem 4.3, we conclude that the formula $\forall X.\, \phi[Y = \chi]$ must be false. Hence, $\exists Y \neq \emptyset \forall X.\, \phi$ is false.

($\Leftarrow$) Suppose S_{χ_0} is not F-preferred. Then, there exists a rational set S' such that $\mathbf{F}(S') \subset \mathbf{F}(S)$.

By clause (1) and deontic consistency of S', at most one of $\mathtt{set}(y_i, 0)$ and $\mathtt{set}(y_i, 1)$ is in $\mathbf{Do}(S')$, for every $y_i \in Y$; moreover, by the clause (2), precisely one of them is in S'. Thus, S' includes a choice χ representing a truth assignment to Y. For this choice, the program $\mathcal{P}'$ essentially reduces under rational semantics to the program $\mathcal{P}$ for $\phi[Y = \chi]$ from the proof of Theorem 4.3. (Observe that only clauses (1) and (2) allow to derive status information for any action $\mathtt{set}(y_i, 1)$). It is not hard to see that the set S there is the unique rational status set of $\mathcal{P}$, if a rational status set exists. it follows that for the choice χ, the formula $\forall.\phi[Y = \chi]$ is true. Since $S' \neq S_{\chi_0}$, we have $\chi \neq \chi_0$. Consequently, $(\exists Y \neq \emptyset)\forall X.\, \phi$ is true. ∎

Since the recognition of F-preferred rational status sets gets more complex in the presence of integrity constraints, also the complexity of computing such a status set increases; the increase is one level in the polynomial hierarchy.

Theorem 4.18 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_\mathcal{S}$, computing any F-preferred rational status set of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$ (so one exists), is in* $\mathrm{FP}^{\Sigma_2^P} \cap \mathrm{RP} \cdot \mathrm{FP}_{\|}^{\Sigma_2^P}$ *and hard for both* $\mathrm{F}\Sigma_2^P$ *and* $\mathrm{FP}_{\|}^{\Sigma_2^P}$.

Proof. Modify the algorithm COMPUTE-RATIONAL-SS in Section 3.2.2 as follows. After Step 2, compute the smallest size s of the $\mathbf{F}$-part over all rational status sets of $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$. Then, in the remaining steps, constrain the oracle queries to existence of a rational status set S', $S \subseteq S' \subseteq (GA \setminus \{A\})$, such that $|\mathbf{F}(S')| = S$. Since the oracle queries are in Σ_2^P (cf. Theorem 4.4), the problem is in $\mathrm{FP}^{\Sigma_2^P}$. Membership in $\mathrm{RP} \cdot \mathrm{FP}_{\|}^{\Sigma_2^P}$ follows from the fact that computing a F-preferred rational status set can be easily expressed as a maximization problem (MAXP) as defined in [4], whose instance-solution relation is co-NP-decidable; $\mathrm{RP} \cdot \mathrm{FP}_{\|}^{\Sigma_2^P}$ is an upper bound for such problems [4].

Hardness for $\mathrm{F}\Sigma_2^P$ is immediate from the proof of Theorem 4.4 (compute a rational status set), since each rational status set of the program $\mathcal{P}$ constructed there is F-preferred. The lower bound of hardness for $\mathrm{FP}_{\|}^{\Sigma_2^P}$ can be shown following the line of the proof of Theorem 4.12, where we reduce the computation of the binary string B for instances $I_1, \ldots, I_n$ of the Σ_2^P-complete complement of the recognition problem for F-preferred rational status sets.

We may suppose that the program is $\mathcal{P}'$ from the proof of Theorem 4.17, and that the set S to check over database D is the set S_{χ_0}, which is rational and has the maximal F-part over all rational status sets. Similar as in the proof of Theorem 4.12, we tag databases D_i by introducing a new column T in each table which is added on the left hand side of FDs, and we add for the tag a variable T to each action scheme. Then, the F-preferred rational status sets S of the obtained program $\mathcal{P}''$ over the union $\hat{D}$ of all tagged databases D_i, are given by the union of the F-preferred rational status sets S_i of $\mathcal{P}''$ over each tagged database D_i. Hence, from any F-preferred status set S of $\mathcal{P}''$, the desired string B can be efficiently computed. Moreover, $\hat{D}$ is constructible in polynomial time from $I_1, \ldots, I_n$. It follows that computing a F-preferred rational status set is $\mathrm{FP}_{\|}^{\Sigma_2^P}$-hard, which proves the result. ∎

For action reasoning, we also find a complexity increase through integrity constraints. Similar as in the case of weak status sets, F-preference lifts this problem to the third level of the polynomial hierarchy.

Theorem 4.19 *Let $\mathcal{P}$ be a fixed agent program (where $\mathcal{IC}$ is arbitrary). Then, given an agent state $\mathcal{O}_{\mathcal{S}}$ and a ground action α, deciding whether $\alpha \in \mathbf{Do}(S)$ holds for (i) every (resp., (ii) some) F-preferred rational status set S of $\mathcal{P}$ on $\mathcal{O}_{\mathcal{S}}$ is Π_3^P-complete (resp., Σ_3^P-complete).*

Proof. The membership part is similar as in the case $\mathcal{IC} = \emptyset$ (Theorem 3.30), but we use a Σ_2^P oracle instead of a NP oracle for recognizing a F-preferred rational status set (Theorem 4.17).

For the hardness part of (i), we extend the construction in the proof of Theorem 4.17. We add another block of quantifiers $\forall Z$ in front of the formula Φ, and encode evaluating the formula $\forall Z(\exists Y \neq \emptyset)\forall X.\phi$ as in Lemma 4.16.

The action base and the database is the same, and field $Tag1$ has value 2 for identifying the Z variables. We add to the program $\mathcal{P}'$ the following clauses:

$$\begin{array}{rcl}
\mathbf{F}(\mathtt{set}(X_1,1)) & \leftarrow & \mathbf{Do}(\mathtt{set}(X_1,0)), \mathrm{VAR}(X_1,Y_1,Z_1,2)\\
\mathbf{F}(\mathtt{set}(X_1,0)) & \leftarrow & \mathbf{Do}(\mathtt{set}(X_1,1)), \mathrm{VAR}(X_1,Y_1,Z_1,2)\\
\mathbf{Do}(\mathtt{set}(X_1,1)) & \leftarrow & \neg\mathbf{Do}(\mathtt{set}(X_1,0)), \mathrm{VAR}(X_1,Y_1,Z_1,2)
\end{array}$$

The clauses effect a choice μ from $\mathbf{Do}(\mathtt{set}(z_i,0)), \mathbf{Do}(\mathtt{set}(z_i,1))$, $z_i \in Z$ (representing a truth assignment to Z), which is passed to the rest of the program. Observe that here, different from the choice rules for Y, the F-parts deduced from different choices μ and μ' are incomparable.

Add to the program another rule

$$\mathbf{Do}(\alpha) \leftarrow \mathbf{Do}(\mathtt{set}(X_1,1)), \mathrm{VAR}(X_1,Y_1,Z_1,1),$$

where α is a fresh action with empty precondition and empty Add- and Del-sets, and let $\mathcal{P}''$ be the resulting program.

It holds that $\mathbf{Do}(\alpha) \in S$ for every F-preferred rational status set S of $\mathcal{P}''$ if and only if $\forall Z(\exists Y \neq \emptyset)(\forall X)\phi$ is true.

To see this, notice that the previous rule is applicable in a F-preferred rational status set S if and only if $\{\mathtt{set}(y_i,1) \mid y_i \in Y\} \not\subseteq \mathbf{F}(S)$ holds. Moreover, each candidate S_χ from the proof of Theorem 4.17, extended for the choice μ according to the new rules, contains $\mathbf{Do}(\alpha)$ just if $\chi \neq \chi_0$. It follows that $\mathbf{Do}(\alpha)$ is contained in every F-preferred rational status set of $\mathcal{P}''$ if and only if $\forall Z(\exists Y \neq \emptyset)\forall X.\phi$ is true. This proves Π_3^P-hardness of (i).

For (ii), add the rule $\mathbf{Do}(\beta) \leftarrow \neg\mathbf{Do}(\alpha)$ to $\mathcal{P}''$, where β is an fresh action of the same type as α. It is easy to see that $\mathbf{Do}(\beta)$ belongs to some F-preferred status set of the resulting program $\mathcal{P}^*$ if and only if $\mathbf{Do}(\alpha)$ does not belong to every F-preferred status set of $\mathcal{P}''$, i.e., $\forall Z(\exists Y \neq \emptyset)\forall X.\phi$ is false. Thus, (ii) is Σ_3^P-hard. ∎

5 Conclusion

In Part II of this series of papers, we have investigated the computational complexity of agent programs in our framework. We have focused on programs which apply to agent states under a generalized domain closure assumption, and where the calls to software code accessing the agent state can be evaluated in polynomial time. The computational problems that we have considered range from deciding the consistency of an agent program on a given state to action reasoning under a particular semantics, and include besides recognition of an acceptable status set (i.e., model checking) also the task of actually computing an acceptable status set. While the former are decision problems, the latter is a search problem; precise computational characterizations of such problems in terms of search problem complexity classes has obtained increasing interest more recently, cf. [4, 17].

As we have shown, for positive agent programs various important computational problems on status sets, and in particular all problems considered on rational status sets, can be solved in polynomial time y the algorithms that we have described. On the other hand, for programs with negation, the different semantics also have a different complexity profile, ranging from the first level of the polynomial hierarchy (feasible status sets) up to the third level (weak rational and F-preferred status sets). Loosely speaking, they confirm the intuition that we have to pay a computational price for selecting a refined and epistemically more appealing semantics.

The use of the results that we have established in our analysis is manifold. Firstly, the results of the complexity analysis of the different kinds of status sets proposed in Part I of this series of papers complement the results on the semantical properties obtained in Part I, and help to better assess the pros and cons of the single Sem-status sets. The results may help an agent application designer in her choice of the appropriate semantics for a particular application of our agent framework. The overview tables in Section 2.3 and the discussion there provides a compact reference for this task.

Secondly, the analysis of the sources of complexity which crop up with the different variants of status sets, and how they effect the complexity of computation (Section 2.3.2), provide insight into how particular principles may effect the complexity of decision making in general. Namely, applying a minimization policy such as preferences to solutions, or a similar maximization policy (as present with weak variants of status sets). These insights may be profitable for other researchers developing frameworks in agent decision making.

Thirdly, our results provide evidence for how optimal algorithms for decision making which handle all possible scenarios (i.e., are complete in that respect), may behave in the worst case, and thus give a clue for the design of such algorithms. As discussed elsewhere [7], the level of the polynomial hierarchy at which a problem resides gives us information about which kind of backtracking algorithm is suitable for solving a problem. E.g., for NP-problems a solution can be found by a simple backtracking algorithm, while for Σ_k^P-complete problems in general, *nested* backtracking of depth k is necessary (unless the polynomial hierarchy collapses, which is not expected). The completeness results for the class $\mathrm{FNP}//\log$ that we have established indicate that the computation of particular kinds of status sets can be optimally implemented as a two phase process, in which first an optimization problem is solved and then a status set is computed as usual. Moreover, they provide some evidence that it is not feasible to parallelize these problems to NP-problems (e.g., calls of SAT routines) in polynomial time.

Several issues remain for further work. One such issue is a comparative study of the computational complexity of different agent frameworks. Since the layout and formal underpinnings of various agent frameworks such as those in [24, 5, 21, 1, 16] are quite different, it is not a priori clear how the complexities of these systems should be compared. Another issue, refining the complexity view, is the expressive power in terms of capability to represent decision processes of inherent complexity. The capability of agent programs in that respect may be formally assessed in the spirit of similar concepts for advanced logical database queries languages [8].

A further important issue are tractable fragments of the agent language that we have presented. The characterization of the sources of complexities provides us with detailed information of which effects have to be eliminated in order to arrive at a polynomial time language. In this direction, we are currently investigating *regular agent programs* [9], in which decision making is layered into levels of polynomial complexity. In this context, approximation techniques and heuristics may be useful. This remains for future research.

Acknowledgements. We wish to thank Alex Dekhtyar, Jürgen Dix, Sarit Kraus, Munindar Singh, Terrence Swift, and the referees of this paper for a very close reading of this manuscript, and for the numerous detailed comments they made. These comments have significantly improved the quality of this paper. We have also benefited from discussions with Swati Allen, Piero Bonatti, Steve Choy, Phil Emmerman, Dana Nau, Anil Nerode, Dino Pedreschi, and Jose Salinas.

This work was supported by the Army Research Office under Grants DAAH-04-95-10174, DAAH-04-96-10297, and DAAH04-96-1-0398, by the Army Research Laboratory under contract number DAAL01-97-K0135, by an NSF Young Investigator award IRI-93-57756, and by a DAAD grant.

References

[1] M. Bratman, D. Israel, and M. Pollack. Plans and Resource-Bounded Practical Reasoning. *Computational Intelligence*, 4(4):349–355, 1988.

[2] M. Cadoli. The Complexity of Model Checking for Circumscriptive Formulae. *Information Processing Letters*, 44:113–118, 1992.

[3] Z.-Z. Chen and S. Toda. The Complexity of Selecting Maximal Solutions. In *Proceedings of the 8th IEEE Structure in Complexity Theory Conference*, pages 313–325, 1993.

[4] Z.-Z. Chen and S. Toda. The Complexity of Selecting Maximal Solutions. *Information and Computation*, 119:231–239, 1995.

[5] P. Cohen and H. Levesque. Intention is Choice with Commitment. *Artificial Intelligence*, 42:263–310, 1990.

[6] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and Expressive Power of Logic Programming. In *Proceedings of the Twelth IEEE International Conference on Computational Complexity (CCC '97)*, pages 82–101, 1997.

[7] T. Eiter, G. Gottlob, and N. Leone. Abduction From Logic Programs: Semantics and Complexity. *Theoretical Computer Science*, 189(1-2):129–177, December 1997.

[8] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive Datalog. *ACM Transactions on Database Systems*, 22(3):364–417, September 1997.

[9] T. Eiter and V. Subrahmanian. Regular Agent Programs and Their Implementation. In preparation, 1998.

[10] T. Eiter, V. Subrahmanian, and G. Pick. Heterogeneous Active Agents. Technical Report UMCP-CSD CS-TR-3880, Computer Science Department, University of Maryland, College Park, Maryland, March 1998. iii + 114 pp. Also IFIG-RR 9802, University of Gießen.

[11] S. Fenner, S. Homer, M. Ogihara, and A. Selman. Oracles That Compute Values. *SIAM J. Comput.*, 26(4):1043–1065, 1997.

[12] M. Garey and D. S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.

[13] G. Gottlob. Complexity Results for Nonmonotonic Logics. *Journal of Logic and Computation*, 2(3):397–425, June 1992.

[14] G. Gottlob, N. Leone, and H. Veith. Second-Order Logic and the Weak Exponential Hierarchies. In J. Wiedermann and P. Hajek, editors, *Proc Conf. Mathematical Foundations of Computer Science (MFCS '95)*, LNCS 969, 66–81, Prague, 1995. Full paper CD/TR 95/80, Information Systems Department, TU Wien, 1995.

[15] J. Halpern and M. Vardi. Model Checking vs. Theorem Proving: A Manifesto. In *Proceedings Second International Conference on Principles of Knowledge Representation and Reasoning (KR-91)*, pages 325–334, 1991.

[16] K. Hindriks, F. de Boer, W. van der Hoek, and J. Meyer. Formal Semantics of an Abstract Agent Programming Language. In *Intl. Workshop on Agent Theories, Architectures, and Languages*, pages 204–218, Providence, RI, 1997.

[17] B. Jenner and J. Toran. The Complexity of Obtaining Solutions for Problems in NP and NL. In A. Selman, editor, *Complexity Theory: A Retrospective II*. Springer, 1998.

[18] D. S. Johnson. A Catalog of Complexity Classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 2. Elsevier Science Publishers B.V. (North-Holland), 1990.

[19] P. Liberatore and M. Schaerf. The Complexity of Model Checking for Belief Revision and Update. In *Proceedings AAAI-96*, pages 556–561, 1996.

[20] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[21] A. Rao and M. Georgeff. Modeling Rational Agents within a BDI-Architecture. In J. F. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the International Conference on Knowledge Representation and Reasoning*, pages 473–484, Cambridge, MA, 1991. Morgan Kaufmann.

[22] R. Reiter. Equality and Domain Closure in First-Order Databases. *Journal of the ACM*, 27(2):235–259, 1980.

[23] A. Selman. A Taxonomy of Complexity Classes of Functions. *Journal of Computer and System Sciences*, 48:357–381, 1994.

[24] Y. Shoham. Agent Oriented Programming. *Artificial Intelligence*, 60:51–92, 1993.

[25] J. D. Ullman. *Principles of Database and Knowledge Base Systems*. Computer Science Press, 1989.

[26] M. Vardi. Complexity of Relational Query Languages. In *Proceedings 14th STOC*, pages 137–146, San Francisco, 1982.

A Appendix: SAT Problems and Quantified Boolean Formulas

The classical satisfiability problem (SAT) is, given a conjunction $\phi = \bigwedge_{i=1}^{m} C_i$ (i.e., a set) of propositional clauses such that each clause C_i is a disjunction $C_i = L_{i,1} \vee \cdots \vee L_{i,n_i}$ of literals $L_{i,j}$ over propositional variables $X = \{x_1, \ldots, x_n\}$, decide whether ϕ is satisfiable. SAT is a well-known NP-complete problem. This remains true if we assume that each clause C_i contains three literals, and either all literals $L_{i,j}$ are positive, or all $L_{i,j}$ are negative; this restriction is known as *monotone* 3SAT (M3SAT) [12].

The dual problem, M3DNF, is complete for co-NP. An instance of M3DNF is a formula $\phi = \bigvee_{i=1}^{m} D_i$ in disjunctive normal form (DNF) which is the negation of an M3SAT instance $\phi' = \bigwedge_{i=1}^{m} C_i$ in CNF (obtained by applying De Morgan's rule). The problem is deciding whether ϕ is a tautology.

We use the following notation. Let ϕ be a propositional formula, and let χ be a truth assignment to the variables in a set of propositional variables Y. (In many places, in abuse of notation χ is a choice representing a truth assignment.) Then, $\phi[Y = \chi]$ denotes the formula obtained by substituting in ϕ for every $y_i \in Y$ its truth value according to χ. Furthermore, $\phi[Y = \emptyset]$ stands for $\phi[Y = \chi]$ where χ assigns *false* to every $y \in Y$.

For example, consider $\phi = x_1 \wedge (\neg y_1 \vee y_2 \vee x_2) \wedge y_1$. Then, for the assignment χ to $Y = \{y_1, y_2\}$ such that $\chi(y_1) =$ *true*, $\chi(y_2) =$*false*, the formula $\phi[Y = \chi]$ is $x_1 \wedge (\neg$*true* $\vee$ *false* $\vee x_2) \wedge$ *true*.

A *quantified Boolean formula* (QBF) is a generalized propositional formula, in which each propositional variable x_i ranges over $\{\ true,\ false\ \}$ and is governed either by an existential ($\exists$) or a universal ($\forall$) quantifier. The truth value of such a formula is obtained by eliminating all quantifiers in the obvious way and evaluating the resulting variable-free formula.

For example, $\forall y_1, y_2 \exists x_1, x_2 . x_1 \wedge (\neg y_1 \vee y_2 \vee x_2) \wedge y_1$ is a QBF. This formula evaluates to *false*, since on assigning e.g. *false* to both y_1 and y_2, the remaining formula is unsatisfiable.

Evaluating a given QBF Φ is a classical PSPACE-complete problem. Syntactic restrictions on Φ provide problems complete for the Σ_k^P and Π_k^P classes of the polynomial hierarchy. In particular, deciding whether a QBF of the form $\exists Y^1 \forall Y^2 \cdots Q_k Y^k.\phi$, where the Y^i are sets of variables and the quantifiers Q_i in front of them alternate, evaluates to *true* is a well-known Σ_k^P-complete problem. Dually, deciding whether a QBF of form $\forall Y^1 \exists Y^2 \cdots Q_k Y^k.\phi$ evaluates to true is Π_2^P-complete. The problem remains Σ_k^P-hard (resp., Π_k^P-hard), even if the quantifier-free part ϕ is, depending on the innermost quantifier Q_k, in M3SAT form if $Q_k = \exists$, and in M3DNF form if $Q_k = \forall$.

Thus, e.g. evaluating a given QBF $\forall Y \exists X.\phi$, where ϕ is in M3SAT form, is Π_2^P-complete. Notice that M3SAT and M3DNF are special cases of QBF (where $k = 1$).

The above problems on QBFs also provide complete problems for the search class counterparts of Σ_k^P. Computing a truth assignment χ such that $\forall Y^2 \exists Y^3 \cdots Q_k Y^k.\phi[Y^1 = \chi]$ is true (i.e., computing an assignment for Y^1 witnessing that $\exists Y^1 \forall Y^2 \cdots Q_k Y^k.\phi$ is true) is complete for $\mathrm{F}\Sigma_k^P$, for $k \geq 1$. Again, hardness holds for ϕ in M3SAT (resp., M3DNF) form.

For the class FNP//$\log$, few natural complete problems are known. An important onw is the problem X-maximal model: Given a SAT instance ϕ and a subset X of its variables, compute the X-part of a model M of ϕ such that $M \cap X$ is maximal, i.e., no model M' of ϕ exists such that $M' \cap X \supset M \cap X$, where a model M is identified with the set of atoms true in it. Completeness of this problem for FNP//$\log$ is shown in [3, 4].[3] Observe that computing a maximum clique in a graph (considered in Section 2.2.2) is not known to be FNP//$\log$-complete.

B Appendix: Notation and Different Kinds of Status Set from Part I

This appendix provides some notation and, for the reader's convenience, the definitions of the various kinds of status sets from Part I which we analyze here.

Definition B.1 (status set) A *status set* is any set S of ground action status atoms over $\mathcal{S}$. For any operator $Op \in \{\mathbf{P}, \mathbf{Do}, \mathbf{F}, \mathbf{O}, \mathbf{W}\}$, we denote by $Op(S)$ the set $Op(S) = \{\alpha \mid Op(\alpha) \in S\}$. $\Box$

Definition B.2 (feasible status set) Let $\mathcal{P}$ be an agent program and let $\mathcal{O}_\mathcal{S}$ be an agent state. Then, a status set S is a *feasible status set* for $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$, if the following conditions hold:

$(S1)$ (closure under the program rules) $App_{\mathcal{P},\mathcal{O}_\mathcal{S}}(S) \subseteq S$;

$(S2)$ (deontic and action consistency) $S, \mathcal{O}_\mathcal{S} \models \mathcal{AC}$, and any ground action α satisfies the following:

- *If* $\mathbf{O}\alpha \in S$, *then* $\mathbf{W}\alpha \notin S$

[3]In [4] a form of reduction among maximization problems is used slightly different from the one in [3]. It requires that the transformed instance $f(I)$ must always have solutions, but for any maximal solution S of $f(I)$, the function $g(I, S)$ is only defined if I has solutions; our proofs of FNP//log hardness can be easily adapted for this setting.

- *If* $\mathbf{P}\alpha \in S$*, then* $\mathbf{F}\alpha \notin S$
- *If* $\mathbf{P}\alpha \in S$*, then* $\mathcal{O}_\mathcal{S} \models Pre(\alpha)$ *(i.e.,* α *is executable in the state* $\mathcal{O}_\mathcal{S}$*).*

$(S3)$ (deontic and action closure) $S = DCL(S)$ and $S = ACL(S)$;

$(S4)$ (state consistency) $\mathcal{O}'_\mathcal{S} \models \mathcal{IC}$, where $\mathcal{O}'_\mathcal{S} = apply(\mathbf{Do}(S), \mathcal{O}_\mathcal{S})$ is the state which results after taking all actions in $\mathbf{Do}(S)$ on the state $\mathcal{O}_\mathcal{S}$. □

Definition B.3 (groundedness; rational status set) A status set S is *grounded*, if no status set $S' \neq S$ exists such that $S' \subseteq S$ and S' satisfies conditions $(S1)$–$(S3)$ of a feasible status set.

A status set S is a *rational status set*, if S is a feasible status set and S is grounded. □

Definition B.4 (reasonable status set) Let $\mathcal{P}$ be an agent program, let $\mathcal{O}_\mathcal{S}$ be an agent state, and let S be a status set.

1. If $\mathcal{P}$ is a positive agent program, then S is a *reasonable status set* for $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$, if and only if S is a rational status set for $\mathcal{P}$ on $\mathcal{O}_\mathcal{S}$.
2. The reduct of $\mathcal{P}$ w.r.t. S and $\mathcal{O}_\mathcal{S}$, denoted by $red^S(\mathcal{P}, \mathcal{O}_\mathcal{S})$, is the program which is obtained from the ground instances of the rules in $\mathcal{P}$ over $\mathcal{O}_\mathcal{S}$ as follows.

   (a) First, remove every rule r such that $B^-_{as}(r) \cap S \neq \emptyset$;

   (b) Remove all atoms in $B^-_{as}(r)$ from the remaining rules.

   Then S is a *reasonable status set* for $\mathcal{P}$ w.r.t. $\mathcal{O}_\mathcal{S}$, if it is a reasonable status set of the program $red^S(\mathcal{P}, \mathcal{O}_\mathcal{S})$ with respect to $\mathcal{O}_\mathcal{S}$. □

Definition B.5 ($A(S)$) For any status set S, denote $A(S) = \mathbf{Do}(S) \cup \{\alpha \mid \alpha \notin \mathbf{O}(S)\}$. □

Definition B.6 (A-relativized action closure) Let S be a status set, and let A be a set of ground actions. Then, the action closure of S under regimentation relativized to A, denoted $ACl_A(S)$, is the closure of S under the rules

$\mathbf{O}\alpha \in S \Rightarrow \mathbf{Do}\alpha \in S$, for any ground action $\alpha \in A$

$\mathbf{Do}\beta \in S \Rightarrow \mathbf{P}\beta \in S$, for any ground action β.

A set S is action closed under regimentation relativized to A, if $S = ACl_A(S)$ holds. □

Definition B.7 (A-relativized status sets) Let $\mathcal{P}$ be a program, let $\mathcal{O}_\mathcal{S}$ be an agent state, and let A be a set of ground actions. Then, a status set S is A-feasible (resp., A-rational, A-reasonable), if S satisfies the condition of feasible (resp., rational, reasonable) status set, where the action closure ACl is replaced by the relativized action closure $ACL_A(S)$ (but DCl remains unchanged). □

Notice that $ACl = ACl_{GA}$, where GA is the set of all ground action atoms.

Definition B.8 (weak rational, reasonable status sets) A status set S is weak rational (resp., weak reasonable), if there exists an A such that S is A-rational (resp., A-reasonable) and there are no $A' \neq A$ and S' such that $A \subseteq A'$ and S' is an A'-rational (resp., A'-reasonable) status set. □

Definition B.9 (F-preference) Let Sem be a kind of status sets. Then, a status set S is an F-*preferred* Sem-status set, if it is a Sem-status set and there exists no other Sem-status set S' which has a smaller forbidden part than S, i.e., $\mathbf{F}(S') \subset \mathbf{F}(S)$ holds. □

C Appendix: Proof of Lemma 4.16

Lemma 4.16 *Let $\Phi' = \forall Z \exists Y' \forall X' \phi'$ be a QBF such that ϕ' is in DNF, where Z may be void. Then, a formula $\Phi = \forall Z \exists Y \forall X.\, \phi$, where ϕ is in M3DNF is constructible in polynomial time, such that*

(1) *the formula $\forall Z X.\, \phi[Y = \emptyset]$ is true;*

(2) *Φ' is equivalent to $\forall Z (\exists Y \neq \emptyset) \forall X.\, \phi$.*

Proof. Without loss of generality, Φ' is already monotone. Suppose $Y' = \{y_1, y_2, \ldots, y_n\}$ and let $Y = Y' \cup \{y_0\}$. Consider the formula

$$\forall Z (\exists Y \neq \emptyset) \forall X'.((\neg y_0 \wedge \cdots \wedge \neg y_n) \vee \phi'). \tag{2}$$

This formula is clearly equivalent to Φ'. Construct next the formula

$$\forall Z (\exists Y' \neq \emptyset) \forall X'.[(\forall W.\psi) \vee \phi'], \tag{3}$$

where $W = \{w_1, \ldots, w_n\}$ and $\psi = (\neg y_0 \wedge w_1) \vee (\neg w_1 \wedge \neg y_1 \wedge w_2) \vee (\neg w_2 \wedge \neg y_2 \wedge w_3) \vee \cdots \vee (\neg w_{n-1} \wedge \neg y_{n-1} \wedge w_n) \vee (\neg w_n \wedge \neg y_n)$. The formula (3) is equivalent to (2). Indeed, observe that $\forall W.\psi$ is equivalent to $\neg y_0 \wedge \cdots \wedge \neg y_n$. To see this, suppose first $\forall W.\psi$ is true. Then, for every $i = 0, \ldots, n$ consider a value assignment χ_i to W such that $w_j = 0$, for every $j \leq i$, and $w_j = 1$, for all $j > i$. Then, $\psi[W = \chi_i] \leftrightarrow \neg y_i$. Hence, $\neg y_0 \wedge \cdots \wedge \neg y_n$ is true. Conversely, suppose $\neg y_0 \wedge \cdots \wedge \neg y_n$ is true. Towards a contradiction, suppose $\psi[W = \chi]$ is false for some truth assignment χ to W. Hence, by the first disjunct in ψ, w_1 is false in χ, which means by the second disjunct that w_2 is false in χ, ..., that w_n is false in χ. However, the last disjunct in ψ is $\neg w_n \wedge \neg y_n$. Thus, this disjunct, and hence $\psi[W = \chi]$ is true, which is a contradiction.

By elementary quantifier pulling (no variable from $Z \cup X'$ occurs in ψ), formula (3) is equivalent to

$$\forall Z (\exists Y \neq \emptyset) \forall X' W (\psi \vee \phi'), \tag{4}$$

and $\forall Z X' W (\psi \vee \phi)[Y = \emptyset]$ is true. By using further universally quantified variables $W' = \{w'_1, \ldots, w'_n\}$, we fleet ψ into M3DNF form and obtain

$$\forall Z (\exists Y \neq \emptyset) \forall W W' [\bigvee_{w_i \in W} (w_i \wedge w'_i) \vee \bigvee_{w_i \in W} (\neg w_i \wedge \neg w'_i) \vee \psi(W/\neg W') \vee \phi'] \tag{5}$$

where $\phi(W/\neg W')$ means the obvious substitution of W' literals for W literals such that the formula is monotone. Clearly, (5) is equivalent to (4). By replicating literals in disjuncts, the lemma is proven. ∎