Decodyn: Treating Hard Problems with Decomposition and Dynamic Programming

Stefan Woltran

September 20, 2012

Contents

Remark. With a previous version of the present project proposal I was one of the 16 finalists for the FWF START price 2011. In the last step of the competition, i.e. the hearing in front of the interdisciplinary jury (see `http://www.fwf.ac.at/de/portrait/jury.html`*), seven projects finally have been selected, none of them stemming from the area of computer science. The current proposal is an updated and improved version.*

1 Overview and Outline of the Project

The proposed research aims at exploring novel ways for efficiently solving problems which involve *huge data* and *complex reasoning* over this data. Such problems provide structure on two levels (data and query), in particular in real-world domains. The main innovation of the project is the development of dedicated methods, based on decomposition and dynamic programming, such that these two levels are jointly exploited. In what follows, we first give an elaborated motivation for our research and then outline its ground-breaking nature. After reviewing the scientific background, we shall formulate the concrete project goals.

1.1 Motivation and Vision

From the very beginning, computers were considered as tools to handle large amount of data (e.g. databases in business domains) or to perform complex reasoning tasks (e.g. planning, expert-systems, diagnosis). But in the world today we face numerous challenging problems which involve the combination of both issues, i.e. complex reasoning over large data. As one example consider social networks such as Facebook which possess hundreds of millions of users, each of them having friendship relations to other users. A typical reasoning problem over such a structure would be to identify a minimal selection of users such that all users are reached via the friendship relation. Similar situations are apparent in various areas, for instance in medicine, where querying ontologies like SNOMED-CT (which contains 311,000 hierarchically organized concepts) is required or in bio-informatics, where complex structures such as proteins or genomes have to be analyzed.

Handling computationally complex queries over huge data is an insurmountable obstacle for standard algorithms, thus alternative methods are required. One solution is to exploit structure. In reality, social networks are not random graphs and medical ontologies are not arbitrary sets of relations. Thus, structural parameters can serve as a key to apply techniques as decomposition (which divides the problem into smaller pieces making use of structural properties of the instance) and dynamic programming which solves the problem in terms of the decomposed instance (see e.g. [13] for an overview on this approach). Notably, *structure* is not solely found in *data* like social networks but is also present in *queries* for complex reasoning (for instance, when seeking for a certain structure in molecules, this is naturally reflected in the query). A prototypical example are so-called query networks [119] in which each node in a social network owns a query, and all of them have to be jointly evaluated.

The main vision of the **decodyn** project is to understand how structure can be simultaneously exploited on these two different levels, in such a way that (**deco**)mposition and (**dyn**)amic programming provide new efficient methods to make complex problems of this kind amenable to today's computers. Several query languages which allow formalizations of complex reasoning tasks are nowadays available. In the course of the project, we will first focus on Answer-Set Programming (also known as disjunctive datalog) [8, 9, 16, 68, 69, 96, 100, 109]. In fact, Answer-Set Programming has successfully been applied in some of the aforementioned application areas (see e.g. [65]) but its use is still limited when problems of huge size have to be solved [56]. In the second phase, we intend to migrate our results and methods to other reasoning paradigms, such as description logics and datalog +/-. As well, we will apply out methodology to concrete problems from the area of bio-informatics.

To summarize, many problems in current applications require powerful query languages capable of reasoning over relations in social networks, analyzing molecular structures, or evaluating ontologies. From a computational point of view, two severe problems arise: First, evaluating such queries is intractable, meaning that there are no general efficient algorithms. Secondly, the amount of data might be huge, thus running times of such algorithms are beyond any reasonable scale. The decodyn approach is proposed to overcome these shortcomings by exploiting structure which is often present in these application areas.

A Running Example. Parameterized Complexity [33, 34, 51, 53, 108] recently shed light on how intractable problems can be treated efficiently when certain structural parameters are taken into account. Thus, also huge problem instances can be solved, in case the instances possess small such parameters. Fortunately, there is significant evidence that many real-world problems are sufficiently structured and potentially provide small parameters, see e.g. [2, 80, 84, 92, 102, 132]. For illustration, consider the two graphs in Figure 1. In fact, the graph on the left-hand side stems from real-world data, namely from the PI's publication record (taken from Sept 1st, 2011) in years 2007–2011 (vertices are co-authors listed at DBLP[1] and an edge between two persons indicates that there is a paper where these two are joint authors), while the graph on the right-hand side is randomly generated with the same number of vertices and respectively edges as the graph on the left-hand side. Although at a first glance the graph on the left-hand side does not look much more structured than the random graph, certain structural parameters are significantly smaller for the left graph. We will come back to this point later.

1.2 Ground Breaking Nature

Thanks to the methods from Parameterized Complexity, NP-complete problems like Constraint Satisfaction Problems (CSP), Bayesian Networks and graph problems are nowadays well understood; in

[1] `http://www.informatik.uni-trier.de/~ley/db/`.

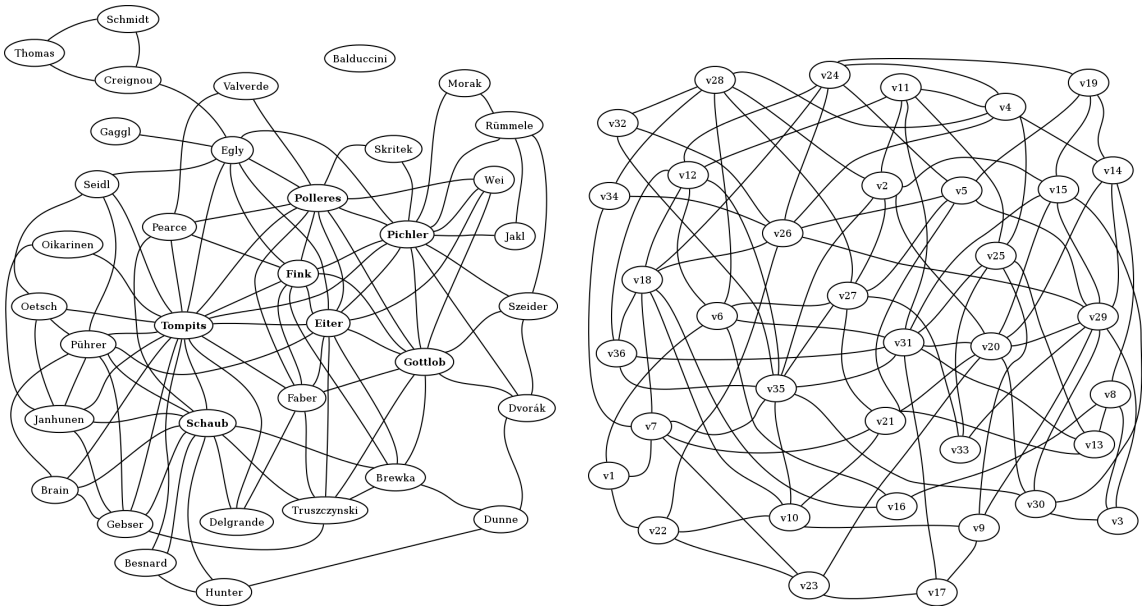

Figure 1: Real-world vs. random graph of same size.

fact, tremendous progress was achieved over the last 20 years, in particular in the area of propositional satisfiability (SAT) where interesting relations between parameters and practical SAT solving have recently been addressed in [5]. It is thus timely to investigate the potential of methods from Parameterized Complexity for more involved reasoning tasks. The innovative nature of the proposed project is to use these methods for highly expressive declarative query languages. Since many problems can be expressed within such languages, the actual design of algorithms (which make use of structure and parameters) is thus *delegated from the human to a system*.

One crucial observation in this context is that declarative specifications of complex reasoning problems naturally *provide structure* (and thus an access point for parameter-driven methods) on *two levels*:

- Data-level: real-world data usually provides some internal structure (e.g. social networks typically show some dense regions of peer groups which themselves are only loosely connected to each other).

- Query-level: the more complex the reasoning tasks to perform are, the more internal structure they provide (this is for instance witnessed by results which analyze the structure of control graphs of code written in imperative languages, see e.g. [126]).

These observations lead us to the following approaches: First, we are interested how structural information of data and queries can be jointly exploited to optimize existing systems. As a second step, we apply the lessons learned in the first step to design completely novel methods which are *directly* based on structural properties and thus go far beyond the current state of art.

Case Studies. The prototypical formalism we are going to focus on in the first phase of the project is *Answer-Set Programming* (ASP). This is a highly developed query language which allows direct formalizations of many of the applications we are interested in. Indeed, the next step the ASP community has to take right now is to offer special-tailored solutions for large-scaled problems.

The reasons why we consider ASP a suitable candidate for our research programme are:

- ASP is well understood in terms of its expressibility [28, 45] and several mature systems (see e.g. [60, 95, 123]) for practical experiments and comparison exist.
- ASP has proven to be successful in many application areas (see e.g. [47, 50, 65, 87]).
- The typical ASP solving architecture separates two aspects of query evaluation (typically called grounding[2] and solving). This provides several access points to exploit structure.
- Relations between ASP and other query languages have already been studied in the literature; this should ease migration of our achievements from ASP to other reasoning paradigms.

The lessons learned in the first phase of the project will pave the way towards the application of our methods to other fields (which we discuss below in the section on "other reasoning paradigms") and selected concrete applications. This will be done in the second phase of the project.

1.3 Scientific Background

Answer-Set Programming

Answer-Set Programming (ASP, for short) [16, 100, 109], also known as A-Prolog [9, 67], is nowadays a well acknowledged paradigm for declarative problem solving. The success of ASP is witnessed by a wide range of applications in the areas of Artificial Intelligence (AI) and Knowledge Representation & Reasoning (KRR) but also in different fields such as bio-informatics [47, 50, 65] or linguistics [18].[3] Most prominently, ASP was also used in the diagnosis-software for the NASA Space Shuttle [111].

ASP builds on a *rule-based language* which includes certain features from non-monotonic reasoning (in particular, default negation) and databases (e.g. aggregates or optimization statements, see [19, 48]). In fact, ASP programs can also be understood as logic-based queries for relational databases and are a proper generalization of the well-known datalog approach making ASP a suitable query language for graph-like structures since concepts like reachability are easily expressed.

To quickly illustrate the functioning of ASP queries, consider the following reasoning problem over graphs as depicted in Figure 1. Suppose we ask for a small selection of my co-authors to check a draft of this proposal, and this selection should cover a wide range of expertise. To this end, an edge

[2] Here we understand grounding as the task of reducing a first-order expression to a propositional one being equivalent on a given domain.

[3] See also `http://www.cs.uni-potsdam.de/~torsten/asp/` for an exhaustive list of applications.

between co-authors may be interpreted as an indicator of similar expertise. Thus, in formal terms, we want to select a minimum set S of vertices from a graph, such that all vertices are either in S or adjacent to a vertex in S (indeed, we look here for minimum dominating sets). In terms of ASP, we can describe this exercise using five simple rules.

$$\{select(X)\} \leftarrow vertex(X); \quad ok(X) \leftarrow select(X); \quad ok(Y) \leftarrow select(X), edge(X,Y);$$
$$\bot \leftarrow vertex(X), not\ ok(X); \quad \#minimize\{select(X)\}.$$

Hereby, the first rule "guesses" whether a vertex is in the set or not; the next two rules check the status for each vertex, i.e. a vertex is "ok" if it is in the guessed set or has a neighbor from that set; the penultimate rule ensures that the guess was correct for each vertex, and finally we minimize the number of selected vertices. For our example, we get four answer sets characterizing the different solutions (the minimum number of persons to contact to meet our specification is 13). For illustration, we provided here a direct (fixed) encoding for an NP-complete problem. However, much harder problems (EXPTIME and above) can be reduced to ASP. More precisely, decision problems for (normal) ASP programs as above are NP-complete[4] in terms of data complexity (i.e. the program is fixed) but NEXPTIME-complete in terms of combined complexity; for more detailled results, see [28, 45].

The architecture of standard ASP systems is depicted in Figure 2 which uses our running example on the co-author graph for illustration. The *grounder* takes the data and the (non-ground) rules of the program and then instantiates the rules w.r.t. the domain which is present in the data. This yields the so-called ground program which is free of variables and subsequently proceeded by the *solver* which finally computes the answer sets. The figure also sketches some of the optimizations performed by grounders; for instance, EDB atoms (facts from the data) are tacitly eliminated from rules.

Let us mention at this point that we have three entry points to analyze structure: the data, the program, and the ground program; structural features of the the ground program however will highly depend on structural features of the input data and the program.

Most state-of-the-art systems like lparse/smodels (developed by Niemelä *et al.*, see e.g. [123]) or the gringo/clasp family (Schaub *et al.*, see e.g. [60]) have a strict separation, while the system dlv (by Leone *et al.*, see e.g. [95]) provides a more integrated grounder/solver approach. Nonetheless all systems suffer from the so-called grounding bottleneck which yields an exponential blow-up, even in cases such a blow-up can be avoided in theory [40].

The vitality of the ASP paradigm is witnessed by recent trends extending ASP to several directions: language extensions for set-variables, lists, or function symbols (see e.g. [24, 46]); hybrid systems to combine ASP with Sat-Modulo Theory [110] or with CSP [103]; and modular ASP [29].

[4]Due to the #minimize statement, complexity is in fact slightly highly [123].

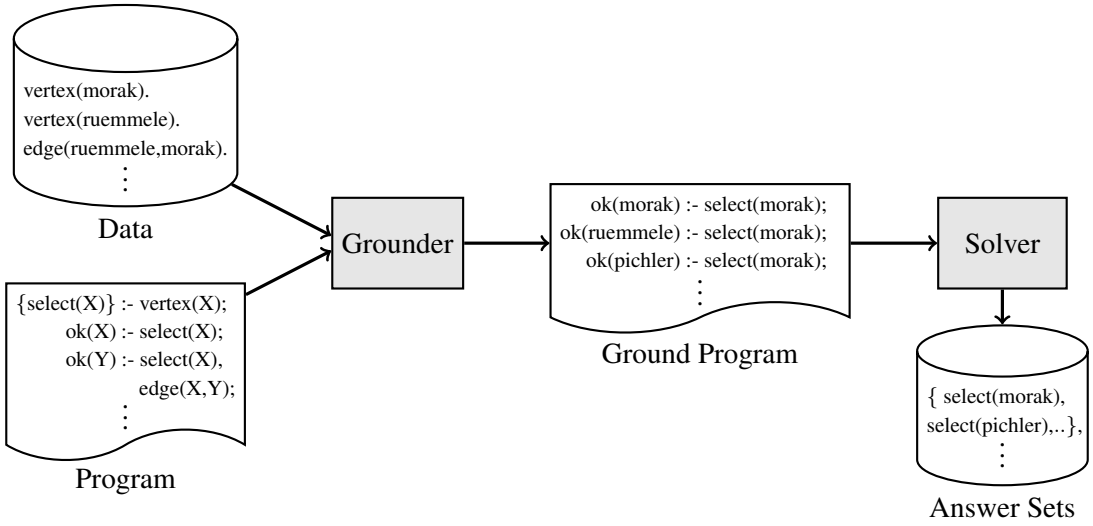

Figure 2: Architecture of an ASP system.

Other Reasoning Paradigms

Querying Descriptions Logics. Description Logics (DLs) form a collection of specially tailored KRR languages, of which most can be seen as decidable fragments of first-order logic [6]. They offer expressive means for modeling and decidable reasoning about knowledge with complex structure. In particular, DLs allow to create *ontologies*—formal conceptualizations of domains of interest—that can be later shared by various task-specific applications. For instance, DLs provide the logical foundation to OWL ontology languages that are meant to describe the resources on the Web in a structured and well understood way [113]. Other application areas include configuration [101], Enterprise Application Integration and Data Integration [94], and more generally Ontology-based Data Access (OBDA) [117]. In the view of OBDA, ontologies consist of two components: the *conceptualization* (or, *schema* in database terminology), and the actual data stored in relations (a database). *Query answering* is a central reasoning task in OBDA, which amounts to answering a database-like query over the ontology, taking into account facts that are not explicitly present but can be logically inferred. The most prominent expressive query language for accessing ontologies is *conjunctive queries*, for which a variety of methods have been developed. Due to the high computational complexity of conjunctive query answering in DLs, which goes up to 2EXPTIME-hardness, to this date no algorithm has been shown to perform well in practice.

Data Exchange. The field of Data Exchange [49] addresses the problem of transforming data from a source schema to a target schema. This is formalized via so-called schema mappings which describe the involved schemes together with constraints the transformed data has to satisfy. Typical such constraints are source-to-target tuple generating dependencies, target tuple generating dependencies and target equality generating dependencies. The core problem of Data Exchange is to decide whether for a given instance of the source schema and a given (or fixed) schema mapping, a suitable instance

of the target schema exists. Similar to ASP, the combined complexity of such problems is very high, while data complexity can remain tractable (see e.g. [90]). Also for this problem, there are two levels where structure can be exploited, namely the data (i.e. the source instance) and the problem description (the schema mapping, in particular the set of constraints).

Datalog +/-. The recently introduced Datalog +/- approach [21, 23] provides a family of ontological languages, where it is possible to represent tuple-generating dependencies and identity constraints using a variant of datalog extended by the use of existentially quantified variables in the head of the rules. Thus datalog +/- is well applicable in the area of data exchange and integration. Also the *DL-Lite* family of description logics can be expressed in datalog +/- [22]. In order to ensure decidability or tractability, certain restrictions are placed on the body of datalog +/- rules. Different such restrictions have been proposed, such as linear, guarded and weakly-guarded datalog +/-. Boolean conjunctive query answering under the latter two restrictions is 2EXPTIME-complete in terms of combined complexity and tractable in terms of data complexity, whereas in linear datalog +/-, PSPACE-completeness is obtained in terms of combined complexity.

Preferential Reasoning. Preferences play a fundamental role in many AI applications. They are needed whenever agents have to choose among alternatives in decision making, or when they have to select a coherent set of beliefs based on possibly conflicting information. For this reason, languages for representing and reasoning with preferences have been developed. One of the best-known examples are CP-nets [15] and generalized forms of CP-nets, see e.g. [73]. These formalisms are of interest within the project, since many decision problems are of high complexity, viz. PSPACE-complete [73]. Moreover, the structure of so-called flips within CP-nets naturally yields a graph, thus structural properties apply to these formalisms as well. Finally, recent approaches [17] proposed rule-based languages for describing conditional preferences, which gives the connection to ASP and similar formalisms.

Parameterized Complexity, Decomposition and Dynamic Programming

Parameterized complexity [33, 34, 51, 53, 108] is a quite novel paradigm to deal with intractable problems. One of the key motivations for this field is that hard problems can become tractable if some problem parameter is bounded by a fixed constant. In particular, problems that can be solved by algorithms that are exponential only in the size of the parameter while polynomial in the size of the input are called fixed-parameter tractable (FPT). One important parameter that often leads to FPT results is treewidth, which measures the "tree-likeness" of a graph and allows for a certain division into subproblems (i.e. the *tree decomposition*) which can then be employed in a *dynamic-programming algorithm*.[5] The intuition behind this idea lies in the fact that many graph problems are

[5]The concept of dynamic programming dates back to the first half of the 20th century. It refers to a method for solving complex problems by breaking them down into simpler subproblems, and to combine the solutions of the subproblems to reach an overall solution.

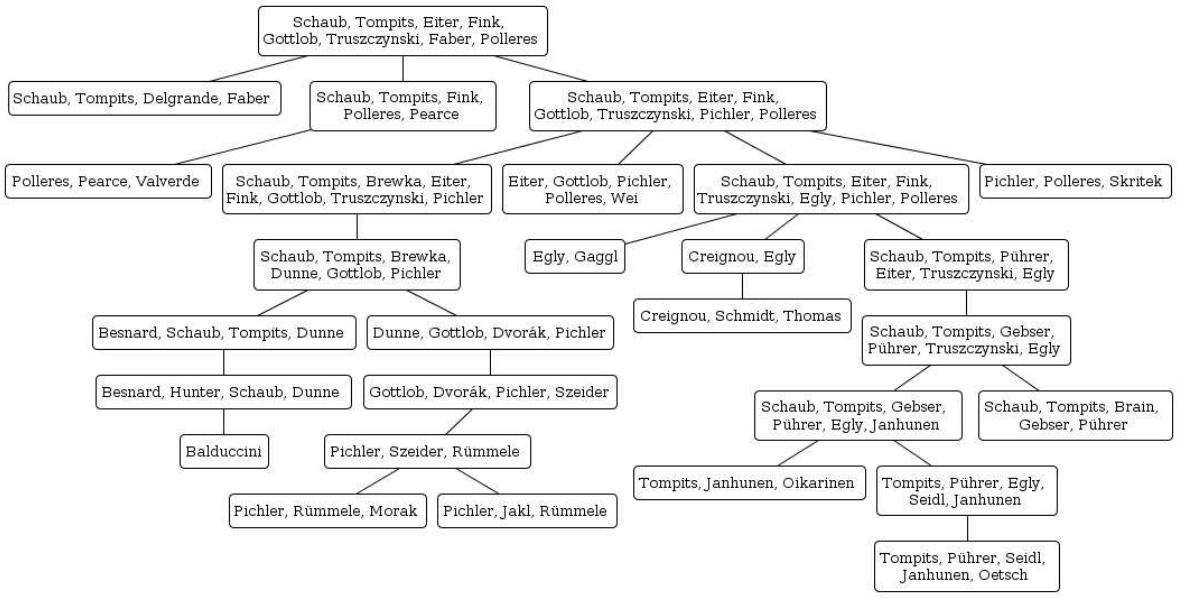

Figure 3: Tree Decomposition of the Co-Author Graph from Figure 1.

easy to solve on trees. Thus, the "closer" a graph is to a tree (i.e. the smaller the treewidth of the graph), the easier the problem becomes to solve. The notions of treewidth and tree decomposition are easily generalized to arbitrary finite structures making them a powerful tool for parameterizing many reasoning problems. Since its introduction by Robertson and Seymour [118], the notion of treewidth has been used to show a huge number of tractability results, see [13] for an overview. Recently, the importance of parameters like treewidth has also been recognized in the areas of bio-informatics (see e.g. [20, 85, 130]) and chemo-informatics (see e.g. [132]).

Formally, a *tree decomposition* of a graph (V, E) is a pair (T, χ), where T is a tree and χ maps each node t of T to a *bag* $\chi(t) \subseteq V$ and satisfies the following properties: (1) for each $v \in V$, there is a node t in T, such that $v \in \chi(t)$; (2) for each $(v, w) \in E$, there is a node t in T, such that $\{v, w\} \subseteq \chi(t)$; (3) for each r, s, t , such that s lies on the path from r to t in T, $\chi(r) \cap \chi(t) \subseteq \chi(s)$. The *width* of a tree decomposition is defined as the cardinality of its largest bag minus one. The *treewidth* of a graph is the minimum width over all tree decompositions of that graph. For a given graph and integer k, deciding whether the graph has treewidth at most k is NP-complete [4], but this problem itself is FPT [12]. As well, good heuristics for estimating the treewidth, and more importantly, for providing tree decompositions of widths not much higher than the actual treewidth of the given graph exist, see e.g. [14, 30]; also complete methods have been developed, e.g. [72].

Coming back to the graphs in Figure 1, it turns out that the real-world graph on the left has treewidth 7; a tree decomposition of this graph of width 7 is depicted in Figure 3. On the other hand, random graphs with the same number of vertices and respectively edges have an average treewidth of around 12 (the particular random graph in Figure 1 has treewidth 13).

To summarize, the efficiency of a combined decomposition/dynamic-programming approach (the

decodyn approach) is based on the fact that (a) efficient heuristics to obtain tree decompositions of low width exist and (b) the running time of the dynamic-programming algorithms mainly depends on the width of the found tree decomposition rather than on the input size. Thus for problems of huge size but low treewidth such an approach has high potential to outperform standard methods. Nonetheless, this method can be applied to arbitrary problem instances making it a general solving method (with a rather different performance behavior compared to standard algorithms, since runtime is now mainly depending on the width of the obtained tree decomposition).

Finally, we briefly discuss results in the area of ASP which are related to parameters: In terms of ground ASP, some FPT results were obtained by Gottlob *et al.* [76] using the seminal meta-theorem by Courcelle [26]. Dynamic Programming for ASP was studied in recent work by the PI [11, 88, 105, 106, 107, 114]. Parameters different from treewidth which have been analyzed are solution size [127], the dimension of feedback vertex sets of a program's dependency graph [78], and the number of loops [98]; also related is recent work on backdoors for ASP [52]. In the non-ground case, classical results on conjunctive queries (see e.g. [75] and the references therein) are of importance; also predicate arity [40, 41] and the number of variables per rule [129] are relevant parameters.

Some Observations

Considering the state-of-the-art, at least the following two shortcomings, which we shall overcome in the proposed project, are identified.

1. So far, application-driven optimization methods for ASP-solvers have mainly focused on heuristics for the solving process which been obtained by learning techniques, see e.g. Balduccini [7]; *the more general approach of exploiting structure in data and rules has not been systematically considered yet.* A recent exception is [10] which suggests to make use of decomposition methods for ASP in a distributed environment.

2. The method of decomposition and dynamic programming has been mainly employed for concrete intractable reasoning problems exploiting structure in the problem *instance*. On the other hand, for problems like conjunctive query most approaches focus exclusively on the *query* structure. *The potential impact of the decodyn approach when applied simultaneously to the problem instance and the problem description has so far remained unexplored.*

1.4 Project Goals

This leads us to the main goals of the project:

- A deep understanding how structural properties can be exploited to improve the performance of existing systems with focus on the concept of tree decomposition and dynamic programming.

- Design of novel systems which build mainly on parameters like treewidth exploiting such parameters on different levels (in particular, in both data and queries).

- Due to the very nature of our approach (which has to deal with partial solutions in the dynamic programming algorithms), we expect that our results are of relevance for alternative solving strategies, for instance parallel methods and query compilers.

In the first phase of the project, we concentrate on the ASP paradigm. In the second phase of the project, we will migrate the successful methods from the first phase to other reasoning paradigms. Let us note that our goal is *not* restricted to develop FPT algorithms for ASP or related paradigms. In turn, we are interested in novel ways of integrating these methods in order to tune existing approaches, and to design completely new methods applicable to structured real-world problems of large size.

2 Methodology

Our central method to achieve the project goals is the combination of decomposition and dynamic-programming algorithms (decodyn, for short). In fact, we consider ***decodyn as a medicine to treat reasoning problems which suffer from the symptoms of high intractability***.

For the decomposition part, we will focus on (hyper)tree decompositions. The reason for this choice is mainly due to the fact that efficient heuristics to obtain such decompositions exist and that there is evidence that small treewidth is often present in real world data. As well, dynamic programming has been extensively studied in the literature and several optimization techniques are available.

We shall group the main objectives of our research into four themes:

1. *Prescribing decodyn to existing ASP systems*: A decodyn-based solver for ground programs is developed; in order to improve its applicability we investigate methods to transform programs such that existing grounders yield (a) smaller ground programs and (b) ground programs with reduced treewidth; for these transformations we also exploit decomposition-based techniques.

2. *Synthesizing novel ASP methods with decodyn*: Here, we will identify the shortcomings of existing ASP grounders in terms of the concepts formulated in Theme 1. On the one hand, improvements of existing grounders are considered; on the other hand, completely novel grounding techniques as well as interleaved grounding/solving methods will be investigated.

3. *Medical Attendance — Accompanying Measures*: This research theme is strongly interlinked with the former two. We collect here all work (theoretical and implementation) that has to be done in terms of tree-decomposition algorithms and dynamic programming systems. As well, the fine-tuning of methods which have proven successful in the above themes is part of this research theme.

4. *Side-Effects — Employ learned lessons to other problems*: Finally, we employ the proposed methodology to other hard reasoning problems. Candidates are datalog variants (in particular datalog +/-), Description Logics queries, preferential reasoning, problems from the area of data exchange, and recent advances in ASP (as hybrid or modular ASP systems). Also a commercial application of the decodyn approach is subject of our research.

To summarize, the first two themes focus primarily on ASP and together with the accompanying work formulated in Theme 3 can be seen as the first phase of the proposed project. Theme 4 is reserved for the second phase of the project; here we go beyond ASP and employ the developed methods in other areas. At this stage, we will stay open-minded to consider additional research questions that are within the general scope of the project.

In what follows, we will discuss the concrete objectives of these themes as well as the state-of-the-art and the methodology we consider to reach these subgoals; the concrete subgoals provide synergetic effects for which we will mention the potential collaborations. At the end of the section we briefly discuss our dissemination strategy.

2.1 Research Theme 1: Injection — Prescribing decodyn to Existing Systems

Inspecting Figure 2, we spot three entry points to exploit structure in an ASP system, the non-ground program, the data, and the ground program. Objective 1.1 is devoted to a decodyn-based ASP solver which makes use of the structure of the ground program. In Objective 1.2 we examine the potential of this approach in the non-ground setting, i.e. we explore how structure from data and programs is preserved by state-of-the art grounders. In Objective 1.3 we shall focus on program transformations suitable for our purposes. In particular, pre-processing of non-ground programs taking structure into account is considered here. The final two objectives are concerned with combining the developed methods into a new portfolio ASP system and with a thorough experimental evaluation. When finalizing Research Theme 1, we will have gained sufficient knowledge about the ways structural information can be used to improve existing systems and about the current limits of this approach.

Objective 1.1: Injection on the Ground Level. Today's ASP solvers rely mainly on DPLL-based search procedures, see e.g. [62, 71]. In preliminary work to this project we have proposed an alternative approach [88, 106] and developed a prototype for a decodyn-based solver, called dynASP; see `http://www.dbai.tuwien.ac.at/proj/dynasp/`. So far, we have implemented different dynamic-programming algorithms for the program classes of disjunctive and head-cycle free programs, see [104] for details. Both approaches work on tree decompositions of the so-called incidence graph, as used in [120]. Figure 4 sketches this concept on a small fragment of the ground program from our running example. Note that the incidence graph distinguishes between vertices for rules and atoms and an edge (a, r) is introduced if atom a occurs in rule r. Thus, this graph is always

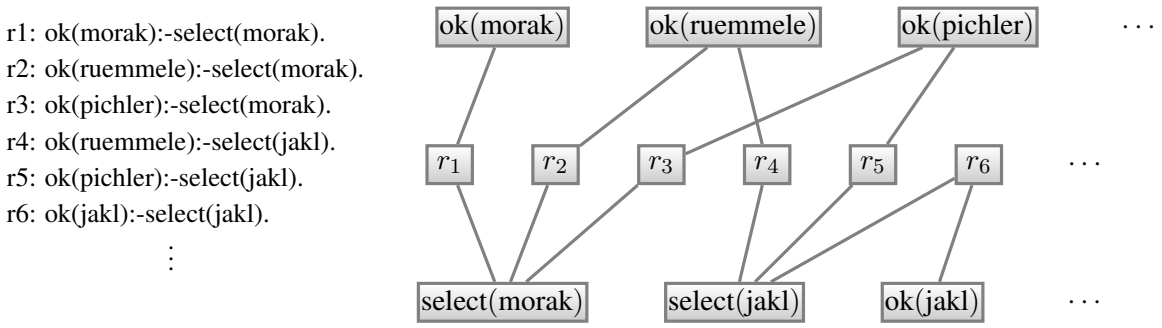

Figure 4: Fragment of a ground ASP program and its incidence graph.

bipartite. In order to improve the current version of dynASP, further algorithms have to be designed and evaluated, in particular for non-normalized tree decompositions [32]. This leads to more involved computations within the dynamic-programming algorithms, which calls for the integration of techniques as used in standard ASP solvers. Also the potential of using further structural information (dependency graph, loops, etc.) has to be explored. Relations to Research Theme 3 (see below) are also apparent. In particular, new techniques for efficient tree decompositions of incidence graphs are required. Special-tailored heuristics for bipartite graphs are thus one direction for research.

To summarize, the goal of this objective is the improvement of the current version of the dynASP solver.

Objective 1.2: Analysis of the Grounding Process. State-of-the-art grounders implement several sophisticated techniques [27, 64] in order to reduce the size of the grounded program. For our purpose, it is more important how treewidth (of the data and/or the non-ground program) is influenced by these techniques. The subject of this objective is thus to understand what treewidth we can expect for the ground program. Clearly, this will highly depend on the class of the non-ground program, i.e. tight or guarded [3, 79] programs are expected to show better behavior in this respect. For instance, the treewidth of the complete incidence graph of the ground program of our running example as obtained by grounders as Gringo is unchanged compared to the treewidth of the data and remains 7. On the other hand, the presence of recursive rules is a severe obstacle to retain structure (consider for instance, the program computes the transitive closure of a graph, then a small treewidth of the original graph is typically lost while grounding). Therefore, we also have to throughly check the literature on datalog where earlier (e.g. [31, 74, 122]) and recent [125] techniques could be of help. In particular, top-down evaluations (as used in the alternative ASP system BProlog) are of interest [134, 135].

The ultimate goal of this objective is thus to understand how structural properties of the input (i.e. from data and rules) are retained in different grounding techniques and in practical systems. This research will be done in *collaboration* with the group of *Prof. Leone (Univ. Calabria)*.

Objective 1.3: Injection on the Non-Ground Level. Here we want to employ the decodyn approach for program rewritings (on both the non-ground and the ground level). One promising approach is the decomposition of rules making use of well known results from conjunctive query evaluation, see [70, 75]. To quickly illustrate this idea consider the rule

$$ok(X) \leftarrow select(X), edge(X,Y), edge(Y,Z), edge(Z,U), edge(U,X).$$

This rule can be split into two rules $ok(X) \leftarrow select(X), edge(X,Y), edge(U,X), help(Y,U)$ and $help(Y,U) \leftarrow edge(Y,Z), edge(Z,U)$, where the $help$ predicate is newly introduced. In general, such a split is obtained by a decomposition of the rule's variable/predicate hypergraph (see [75]). Note that the number of variables per rule is decreased. Due to the cyclic dependency of the variables, current grounders do not yield any optimization here (in particular, if $edge$ would be predicate derived in the program). Pilot tests [107] have shown the potential of rule decomposition, where significant speed-ups for instances from the recent ASP competition [25] are documented. Thus, we will continue the research originated in [107]. Another topic is the applicability of equivalence-preserving rewriting techniques (see [61] and the PI's work [42, 43, 44] based on strong equivalence [97]). Furthermore, partial pre-grounding which uses structural features of the data to make the rules more "sensitive" w.r.t. the structure of the data could help grounders to avoid generation of useless rules. Also the usage of the predicate/rule incidence graph on the non-ground level is subject of our research. All these techniques require efficient decomposition methods which again leads to a connection to Research Theme 3.

The ultimate goal of this objective is thus the design of pre-processing techniques, which allow state-of-the-art grounders to produce ground programs of smaller treewidth. This research will be done in *collaboration* with the group of *Prof. Gottlob (Univ. Oxford)*.

Objective 1.4: Next-generation Portfolio ASP-engines. A promising research field to obtain practical efficient systems is the area of algorithm selection (see e.g. [124, 131]) which mainly is devoted to select the most suitable configuration of a solver for a given problem instance. In the area of ASP, such systems were realized only recently [57, 99] but show promising results. Our proposal is to bring algorithm selection to a new level by taking structural properties of the given program and input into account. Feature selection [81] techniques thus have to be studied and applied. In particular, this would allow to select whether a standard solver or the decodyn-solver dynASP is applied on the ground level of ASP.

The ultimate goal of this objective is thus the integration of the novel methods obtained in Objectives 1.1 and 1.3 into a next-generation portfolio approach which selects certain algorithms after inspecting the structure of the input. This research will be done in *collaboration* with the group of *Prof. Schaub (Univ. Potsdam)* who has experience with ASP portfolio systems and with *N. Musliu (TU Vienna)* who is an expert in meta-heuristics and related methods.

Objective 1.5: Experimental Evaluation. Finally, we have to evaluate the suite of systems and tools we have developed in course of these first research steps. We will make use of standard ASP benchmarks [25] but we are also interested in problems which provide more structure, for instance ASP programs generated for (single player) general-game playing problems and PDDL problems [58].

In fact, the goal of Objective 1.5 is to identify application areas where our methods already lead to significant speed-up but we also have to understand the limits and problems of our approach, in particular in connection with existing grounders. This brings us to the second group of objectives.

2.2 Research Theme 2: Surgery — Synthesizing Novel Methods with decodyn

Here, we identify the shortcomings of existing ASP grounders when used in the approaches of Theme 1. On the one hand, improvements of existing grounders are considered; in particular, they should exploit meta-information from pre-processing steps. On the other hand, we consider the use of decomposition to divide the grounding process and apply interleaved solving (again employing existing systems). However, we even want to go beyond this by directly making use of the structures provided by data and/or rules to develop a new system, which follows an integrated grounder/solver concept, thus avoiding grounding bottlenecks whenever possible. Additional objectives address the relation of our approach to recent research in ASP which deals with inherently incomplete programs; the collection of theoretical results in terms of Parameterized Complexity; and an experimental evaluation.

Objective 2.1: Integration. To fully exploit the pre-processing methods developed in Objective 1.3, integrating them to state-of-the-art grounders is necessary. For instance, grounders should use information about helper predicates (as the $help$ predicate in our example) in order to speed up the grounding process. As well, certain information might be even passed to the solver, to make it aware of the tree-like structure of the splitted rules in the grounded instance. Another way to provide our methods to existing systems is the concept of meta-programming, see e.g. [59]. An interesting realization of such an approach in terms of a parameter (predicate arity) has recently been presented in [41]. We also intend to investigate the potential of decodyn-based methods for predicate decomposition (this reduces the arity of predicates, which is also a crucial parameter in terms of performance) or for novel forms of (parameterized) splittings (so far splitting is solely based on components in the dependency graph, but this condition can certainly be relaxed). However, we can go far beyond the integration of pre-processing techniques. In fact, decomposition allows to split the entire answer-set computation into smaller fractions suggesting an interleaved grounding/solving process (which still relies on calls to existing ASP systems) along the lines of a suitable decomposition of the program.

The overall goal of this research objective is to integrate decodyn-based methods to existing systems. This objective is conducted in *collaboration* with the two main ASP groups, i.e. the one of *Prof. Schaub (Univ. Potsdam)* and *Prof. Leone's* group at *Univ. Calabria*.

Objective 2.2: Building a Novel System. The results from Objective 2.1 will also guide us towards a novel ASP system which makes direct use of the structure provided by the data and the program. As a starting point, one can consider the decomposed approach of ASP solving (where the entire program is split to smaller pieces, each of which solved by existing systems). We expect that the communication amount required for propagating solutions of subproblems will yield a certain overhead which can be circumvented by only partially solving the subproblems and propagating fragments of solutions along the decomposition. Also the incorporation of a top-down approach is of interest and calls for novel methods (e.g. certain forms of lazy or on-demand grounding [93, 112]). The bottomline is that the design of a completely novel system will overcome certain shortcomings of the approaches discussed in Objective 2.1. Indeed, such a new system would allow us to integrate all the discussed methods at once, but also to combine them in a more direct way (note that we so far have mostly separated the decodyn-approach on the ground level (Objective 1.1) and on the non-ground level (Objectives 1.3 and 2.1)). In this context, new forms of incidence graphs (which jointly take predicates, rules and variables into account) and according methods for decomposition will be developed. Moreover, recent advances in grounding, see e.g. [1], have to be explored and integrated.

To summarize, depending on the achieved results in terms of decodyn-based methods for solving ground programs and pre-processing non-ground programs, a novel system will be designed which is intended to combine all the features.

Objective 2.3: Side Results. In this research objective, we will investigate how our methods and results relate to approaches of a similar nature, for instance ASP scenarios where incomplete programs play a certain role and systems that put Courcelle's theorem to practice. In fact, due to the very nature of the dynamic-programming approach, our methods will implicitly have to handle program fragments. The concepts of reactive ASP [54], distributed ASP [10] (see also [31, 122] for early results in the world of datalog) and incremental ASP [63] face a similar problem and thus we see interesting connections to our methodology. We also mention that decompositions of programs can be used to *compile* non-ground ASP programs into dynamic-programming based executables which can then be run on varying input data.

This brings us close to meta-theorems *a la* Courcelle; hence, also the relations to the monadic datalog approach [77] and to the MSO-based system by Langer *et al.* [91] have to be investigated. Also to mention here is a novel system called DFLAT [11] developed by our group[6]. Although this system also makes use of tree decomposition and dynamic programming in an ASP environment its original aim is orthogonal to the proposed research. In fact, DFLAT is designed as a rapid prototyping tool for dynamic programming, while in the current project the dynamic programming aspect has to be done by the system itself (similar to the mentioned MSO-based system by Langer *et al.*). Recall that our proposed approach however exploits structure on both data and query level, while the systems

[6] `http://www.dbai.tuwien.ac.at/research/project/dynasp/dflat/`.

mentioned here only decompose on the data level. Nonetheless, tools like DFLAT will be very helpful in course of the project and we expect mutual synergies which will foster the development of the systems within the decodyn project.

Collaborations: *Prof. Pichler (TU Vienna)* and *Prof. Gottlob (Univ. Oxford)*.

Objective 2.4: Theoretical Results. While Objective 2.2 is more focused on practical realization, we aim here for theoretical results which classify the complexity of ASP w.r.t. (a combination of) different parameters. Besides the central concept of treewidth in data and rules, we have to consider here parameters as predicate arity, solution size, number of variables, number of cycles, etc. Also the role of different program classes and guarded fragments have to be investigated. As an ultimate goal of this work package, we aim at FPT results which will definitely require a combination of parameters. In order to obtain a precise picture, also parameterized hardness results for complexity classes in the so-called W-hierarchy will be derived. We plan collaborations with *Prof. Eiter*, *Prof. Szeider* and *Prof. Pichler (all TU Vienna)*.

Objective 2.5: Experimental Evaluation. We also schedule practical experiments for this research theme. In particular, for the novel system developed in Objective 2.2, we shall identify possible applications domains where this new method outperforms standard approaches. This will be also of importance for the decisions in Research Theme 4.

2.3 Research Theme 3: Medical Attendance — Accompanying Measures

To achieve the goals mentioned so far, a number of accompanying measures has to be undertaken. In the following first two objectives we are concerned with the fundamental methods underlying our proposal, namely tree decomposition and dynamic programming. In preliminary work, we already developed a platform (called SHARP) which combines existing tree-decomposition heuristics (using the library by Dermaku *et al.* [30]) with a system that offers several useful features to implement dynamic-programming algorithms. SHARP allows for rapid prototyping which is indispensable in the first steps of the project. Optimizing and extending the SHARP framework is thus a particular item on our list of work packages. The remaining three objectives are concerned with other general aspects we have to deal with, namely feature selection, extensions of decodyn methods to the full amplitude of the ASP language, and the consideration of alternative parameters.

Objective 3.1: Heuristics and Customized Decompositions. This objective includes the study of specialized heuristics for tree decomposition (recall that we have to deal with bipartite incidence graphs); also methods which take the orientation of edges into account are of relevance (for analyzing dependency graphs of programs). As well, the incorporation of recent advances from this area (see e.g. [66, 89]) is required. We also expect that "customized tree decompositions" (where one can

specify some preferences about the location of nodes in the tree decomposition) are of importance for our purposes; for instance, rules which constraint the potential answer sets, are better located down in the tree, in order to avoid the computation of local solutions which are ruled out later anyway. To this end, concepts as weighted tree-decompositions [121] have to be explored and suitably adapted.

Objective 3.2: Algorithm Optimization. Dynamic Programming is at the heart of our methods, thus all relevant results in the literature have to be explored and integrated to our methods. This includes special algorithms for fast joins and sophisticated propagation of partial solutions (see, e.g. [128]). Another interesting option is to exploit functional dependencies in tree decompositions and in the dynamic-programming algorithms, see [133].

The goal of Objectives 3.1 and 3.2 is thus the further development of the SHARP framework which is our central tool to implement the decodyn methods required for our project objectives. *N. Musliu (TU Vienna)* will act as the main *collaborator* in these tasks.

Objective 3.3: Algorithm Selection. This objective is mainly concerned with the development of dedicated meta-heuristics useful for our purposes (as already outlined in Objective 1.4, where we discussed our portfolio approach). However, the concept of feature selection is crucial in many of the methods we propose. As an example, we mention here the results from our preliminary work on decodyn algorithms for ground ASP programs [104], where we observed that the efficiency of different heuristics for the tree decomposition is highly dependant on structural features of the incidence graph of the given ground program. In other words, evaluating the interplay between tree-decomposition heuristics and custom-tailored dynamic-programming algorithms is paramount for efficient solutions. As *collaborators* in these tasks, *N. Musliu (TU Vienna)* and *Prof. Schaub (Univ. Potsdam)* are foreseen.

Objective 3.4: Integral Medicine. The development of methods and tools in course of the Research Themes 1 and 2 focus on a core-fragment of the ASP language (i.e. disjunctive programs). This is because we want to be flexible in coming up with several prototypical systems. For those systems which turn out to be of practical relevance we thus have to extend and generalize our results to the full ASP language offered by state-of-the-art ASP systems. This includes choice rules, weight constraints [123], aggregates [48], minimize/weak constraints [19] and built-in arithmetics. For the latter, we can leave the actual computation to the dynamic-programming algorithm, thus reducing the grounding size. Note that this provides an elegant solution to one of the particular shortcomings of current ASP systems which are not able to efficiently handle (large) integer domains. Collaborators for this work-package: *Prof. Eiter (TU Vienna)* and *Prof. Schaub (Univ. Potsdam)*.

Objective 3.5: Different Therapy. Another objective accompanying the entire Research Themes 1 and 2 is based on the fact that we do not want to limit ourselves to the parameter of treewidth. Indeed,

several similar parameters like cliquewidth, branchwidth, or rankwidth exist [83] and might be also suitable for our project. In particular, graph parameters which take the orientation of edges into account are of relevance when we consider the dependency graph of a program. In addition, we also include here the study on backdoors which allow, roughly speaking, to exploit problem instances which are "close" to easy fragments, see e.g. [52] for recent work in terms of ASP. As a partner for this objective, *Prof. Szeider (TU Vienna)* has agreed to collaborate.

2.4 Research Theme 4: Side-Effects — Employ Learned Lessons to Other Problems

Depending on the results we will have achieved in first three research themes, selecting further formalisms to migrate the decodyn method is at the start of this second project phase. In addition, we intend to apply our methods to a particular problem domain, namely from the area of bio-informatics.

Candidate formalisms we study in Objectives 4.1 and 4.2 have been already outlined in the section on Scientific Background, where we briefly discussed the areas of (querying) Description Logics, Data Exchange, datalog +/-, and preferences. A further natural candidate to consider here are ASP extensions (function symbols, set and list variables, modular ASP approaches) and Hybrid ASP systems. Let us mention at this point certain relations between some of the candidate formalisms, in particular between ASP, datalog +/-, and Description Logics. In fact, ASP follows the principle of Closed-World Reasoning; while DLs follow an Open-World Assumption; datalog +/- however "sits" inbetween, thanks to the concept of existential variables in rule heads (recall that datalog +/- has many DLs as subclasses). But there also direct relations between ASP and DL, for instance [82, 86]. This will help us to apply our results to these other formalisms.

In Objective 4.1, we apply our results from Research Theme 1, i.e. we aim to improve existing systems, while Objective 4.2 is reserved for applications of results from Research Theme 2, i.e. here we use our results and tools to design a system for a formalism which has not been put to practice yet.

Objective 4.1: Migrate Results from Research Theme 1 to Formalism A. Candidate formalisms are: ASP extensions which are already implemented (for instance, systems as DLV complex or DLV-HEX; also function symbols are nowadays supported by ASP systems). For this selection, the most natural *collaboration* would be with the group of *Prof. Eiter (TU Vienna)*. The same holds for querying Description Logics, where systems as RACER can be exploited. Additional local collaborations with *M. Ortiz* and *M. Simkus* (both *TU Vienna*) are considered.

Objective 4.2: Migrate Results from Research Theme 2 to Formalism B. Here, we want to select one of the following formalisms: datalog +/- (collaboration: *Prof. Gottlob, Univ. Oxford*), preferences and CP-nets (collaboration: *Prof. Brewka, Univ. Leipzig* and *Prof. Truszczyński, Univ. Kentucky*), or data exchange (collaboration: *Prof. Pichler*). The goal of the objective is the design of a new system which makes direct use of structural properties following the methods discussed in Objective 2.2.

Objective 4.3: Application in the Bio-Informatics Domain. Compared to Objectives 4.1 and 4.2. which are devoted to formalisms, our final objective is on a particular application area where our methods and tools might provide novel and more efficient solutions. We want to focus on large-scale biological networks, in particular using an abstract representation of such networks via the Sign Consistency Model (SCM). SCM is based on influence graphs which describe cellular interactions. This graph structure makes tree decomposition techniques a suitable method for problems in this field. Real-world instances are available, e.g. representations of E.coli and yeast. Although these graphs are huge, they have relatively low treewidth (E.coli: 22, yeast: 26) and thus exactly meet what we have anticipated for real-world data. Moreover, it is desired to declaratively query such structures. In fact, many different problems are of interest here, for instance repairing such networks [55] with respect to noisy data. The group of Prof. Torsten Schaub in Potsdam, recently applied ASP to such problems [65], but they did not take structural features into account yet. Exploiting structure in this particular domain will extend the range of applicability of declarative query methods to larger networks, potentially providing exciting implications beyond the field of Computer Science. A collaboration with *Prof. Schaub* is thus natural here.

2.5 Dissemination and Deliverables

Dissemination: Besides publications in journals and conferences of high reputation, we will participate with our ASP-solvers (either enhanced by the proposed methods or completely new prototypes) at the bi-annual ASP competitions [25]. The benchmarks used in these competitions cover some of the applications fields we mentioned and thus are a perfect way to evaluate the success of the project.

Deliverables: A Technical Report for each research objective is planned in order to collect the results and to accompany the project progress. At the end of the project an extended survey about the obtained results, tools, and gained insights will be layed out.

3 Further Project Relevant Considerations

3.1 Feasibility and Expertise of the PI

The PI has shown his potential as a group leader in course of a WWTF project on argumentation[7], see `http://www.dbai.tuwien.ac.at/proj/argumentation/`. The success of this project is witnessed by the large number of publications (so far, 8 journal papers and around numerous conference papers). Among them there is work which is related to the research of the proposed project; first, we successfully applied techniques as decomposition and dynamic programming [35, 36, 38] to argumentation; second, also the relation between ASP and argumentation has been considered [37, 39].

[7]The Viennese Science Fund (WWTF) offers thematic calls for research projects similar to projects funded by national science agencies. The PI's proposal was among the projects granted in the 1st ICT call of the WWTF; acceptance rate: 25%.

The PI's general expertise in the area of ASP (see e.g. [39, 40, 42, 43, 44]) as well as on decomposition methods and dynamic programming (see e.g. [36, 115, 116]) provides a solid basis for the challenges of the decodyn project. In particular, we already have applied decomposition and dynamic programming to the grounded version of ASP [88, 104, 106, 114]. In preparation of this project proposal we also did some pilot tests for general ASP preprocessing techniques [107]; see Objective 1.3 for a broader discussion. As well, the already mentioned DFLAT system [11] offers ASP as a declarative tool for rapid prototyping of dynamic programming algorithms on tree decompositions.

3.2 Host Institution and Synergies

The proposed project will be carried out at the Institute of Information Systems of the Vienna University of Technology. The institute hosts five departments (Distributed Systems, Databases & AI, Knowledge-Based Systems, Formal Methods in Systems Engineering, Parallel Computing) and provides a perfect environment to reach the ambitious project goals. Moreover, our scope is nicely embedded within the objectives of several projects currently conducted at the institute, but still significantly differs from those other projects. Several members of the institute will support the applicant in course of the project:

Thomas Eiter is the head of the institute and a leading expert in the area of Knowledge Representation and Reasoning. His research group runs several related projects, in particular about ontologies and ASP (e.g. FWF P20841, FWF P20840, and the ONTORULE project FP7 231875). Our proposed research would provide a perfect interface to these projects. In general, Prof. Eiter's expertise will be very useful to tackle several of the project goals.

Reinhard Pichler is the head of the department where the PI is currently employed (Databases and Artificial Intelligence). Prof. Pichler has a focus on practical realizations of FPT algorithms; in a joint project (FWF P20704), we have developed several dynamic-programming algorithms. These aspects are also central in the proposed research, thus his support will be of great help. The other main focus of Prof. Pichler's group lies in the area of database theory, in particular schema mappings; this expertise will be extremely useful for Research Theme 4, where we listed data exchange as a candidate for applying our results to formalisms beyond ASP.

Stefan Szeider currently runs an ERC Starting Grant Project on "Parameterized Complexity of Reasoning Problems". Its focus is mainly on propositional variants of reasoning and the applicability of different parameters and concepts, while our project is devoted to rule-based formalisms which naturally have to deal with variables, grounding, etc. Prof. Szeider is a well recognized expert in complexity theory and his support will be extremely useful for the theoretical issues of our research.

Post Docs: There are several excellent Post-Docs at the institute which will be integrated into this project. *Nysret Musliu* is an expert in heuristic methods and algorithm selection. His experience will be of great help in Research Theme 3. *Magdalena Ortiz* has contributed central results in the area of

Description Logics, and *Mantas Simkus* has deep knowledge about ASP and recent extensions (e.g. ASP with function symbols). Both are thus potential collaborators for Research Theme 4.

3.3 Collaborations

International Academic Collaborations. International researchers supporting the applicant's programme through close collaborations (detailled contents for the collaboration are mentioned in Section 2) are the following:

- The group of *Prof. Schaub (Univ. Potsdam)* is in charge of today's state-of-the-art systems in ASP. They currently investigate problems as reactive or incremental ASP where we see a high potential for successful joint work. The collaboration with Prof. Schaub is of high importance, allowing us to benefit from the practical experience of this group in the development of ASP systems.

- *Prof. Leone (Univ. Calabria)* and colleagues are the founders of the pioneering ASP system dlv. The distinguished features of dlv are its close grounder/solver interface and the successful integration of methods from the database area. Their expertise is thus of high value for our research goals.

- The group of *Prof. Gottlob (Univ. Oxford)* is a world-leading center for research in the areas of (hyper)tree decompositions, datalog, and many more. In particular, joint research on datalog +/- is planned. The project team will greatly benefit from a close collaboration with Prof. Gottlob's group.

- For the area of preferences, one of the possible formalisms to be studied in Theme 4, we plan to collaborate with *Gerhard Brewka (Univ. Leipzig)* and *Mirosław Truszczyński (Univ. of Kentucky)*.

Note that the PI has already published papers with people from all these groups (concerning Univ. Calabria, joint work has been done with W. Faber), see Figure 1.

3.4 Potential Impact

The research proposed is *interdisciplinary* in the sense that it is located on the interface between purely theoretical research (Complexity, and in particular Parameterized Complexity) and the diverse field of AI and KRR which hosts the area of complex reasoning of which Answer-Set Programming is one of the most important representatives. Also, the area of databases is touched by the planned research.

Potential Impact. The proposed methodology of decomposition and dynamic programming algorithms makes use of structural parameters and thus has potential to bring the studied formalisms (in particular, the paradigm of Answer-Set Programming which we focus on) to a new level of applicability and relevance. This claim is justified by the fact that we consider real-world problems (and their particular structural properties) as those examples which should underly further developments of algorithms and systems. The proposed research is timely and enters unexplored ground.

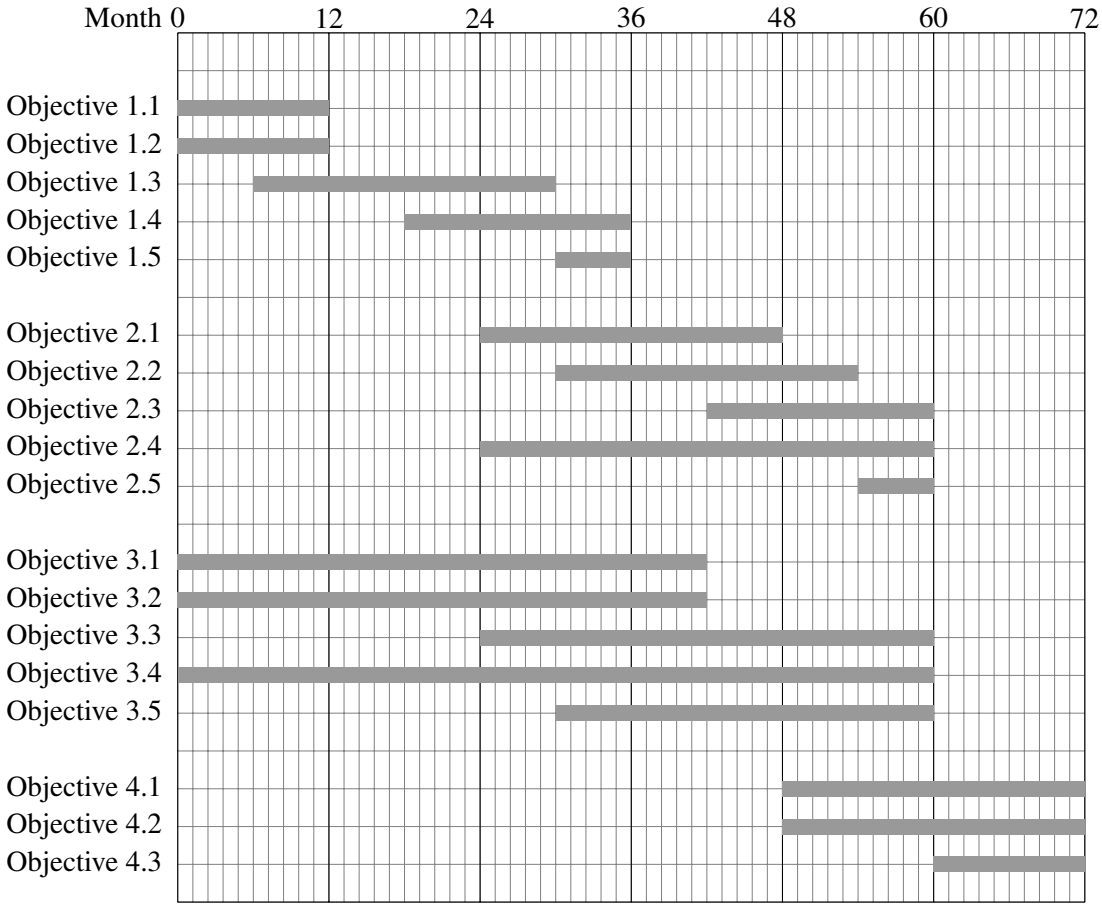

Figure 5: Project Schedule.

High Gain/High Risk. In particular, the synthesis of the studied methods within a completely novel system explores unconventional new methods that are far beyond the current state-of-the-art. This naturally implies a considerable risk. However, this risk is mitigated by the fact that even partial solutions will at least provide optimizations for existing methods and an estimate to what extent current parameter-driven methods are suitable for highly complex reasoning problems.

4 Work Plan and Requested Funding

Timeplan

The chart in Figure 5 sketches the schedule for the project along the lines of the research objectives as detailled in Section 2. Roughly speaking, the first two years are concerned with Research Theme 1, where we will inject our methods to existing systems. In the years 3–5 the integration of the methods to existing systems as well as the design of novel systems are planned. Thus years 1–5 will be concerned with ASP together with the necessary work packages concerning the accompanying measures (Research Theme 3). In the final two years (i.e. years 5 and 6) we enter the second phase of the project,

	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Total
Senior Post-Doc	66.680	66.680	66.680	66.680	66.680	66.680	400.080
Post-Doc	60.610	60.610	60.610	60.610	60.610	60.610	363.660
Pre-Doc1	34.700	34.700	34.700	34.700	34.700	34.700	208.200
Pre-Doc2	0	0	34.700	34.700	34.700	34.700	138.800
Travel	13.000	13.000	15.000	15.000	15.000	15.000	86.000
Total	174,990	174,990	211,690	211,690	211,690	211,690	1.196,740

Table 1: Cost Summary.

where we apply our methods to other reasoning formalisms and also focus on a particular application area. The final year is exclusively reserved for these tasks.

One central milestone arises after the third year where we conclude the first set of objectives. At this point we expect to have sufficient knowledge about the potential of the investigated methods in order to select concrete directions for Research Theme 2 and also to decide about Objectives 4.1 and 4.2, i.e. which further formalisms we shall consider. A second milestone is located at the end of Year 5. At this stage we will have a suite of tools available in order to go towards implementations of systems for the selected formalisms and to tackle the concrete application (Objective 4.3).

Project Costs

Project Personnel. For a successful accomplishment of the ambitious project objectives, funding that covers the salaries of the PI, one Post Doc, and two PhD positions is requested. More precisely, we ask for a total amount of **EUR 1,110,740** for personnel costs, calculated as follows:[8]

Senior Post-Doc. The amount of **EUR 400,080** will cover the full salary of the applicant (his current 6-year contract as assistant professor at the Vienna Univ. of Technology ends with August 2013).

Post-Doc. The wide range of disciplines and topics involved in the project requires an additional researcher at post-doctoral level. Her/his task will be to support the PI in all aspects of the project. The cost for a post-doc over the entire project period is **EUR 363,660.**

PhD students. Funding of PhD students for in total 10 years is requested (**EUR 347,000**). The responsibilities w.r.t. the research objectives (O1.1–O4.3) are partitioned between the two students (S1 and S2) as follows: S1 is associated to Research Theme 1 (O1.1–O1.5); S2 starts in year 3 and is mainly involved in Research Theme 2 (O2.1–O2.5). Concerning Research Theme 3 we consider the following partition: S1 has to focus on the development of the SHARP framework (O3.1 and O3.2) and the implementation of dynamic-programming algorithms, while S2 is in charge of aspects as algorithm selection and meta-heuristics (O3.3) which are important for the portfolio approach, as well as of O3.5. O3.4 is central for both S1 and later S2. The tasks of Research Theme 4 will be split between S1 and S2 according to the workload the other objectives will create.

[8]The costs are given according to the current table at `http://www.fwf.ac.at/de/projects/personalkostensaetze.html`.

Travel and Visits. These costs are calculated such that the project team can make several trips per year for attending conferences or visiting international collaborators (in the first two years, we reserve EUR 7,000; in the remaining four years, when the team is enhanced by one member we reserve EUR 9,000). We also request funding for travel and accommodation of research visitors, allowing for 3 visits per year (EUR 2,000 each). This yields yearly costs of EUR 15,000 and **EUR 90,000** in total.

Total Costs. A total of **EUR 1,196,740** is requested; see Table 1 for an overview.

5 Human Resources and Career Development

My research is located at the interface between the areas of Artificial Intelligence (AI) and Knowledge Representation & Reasoning (KRR) on the one side, and on computational methods on the other side. As an ultimate goal, I consider the application of novel logical and algorithmic methods to important problems in AI and KRR; firstly to understand the inherent computational complexity of these problems, and then secondly, to use these insights to obtain practical solutions which bring such formalisms into a state of real-world applicability. The proposed project provides an exciting opportunity to show my skills in a large-scaled project with practical implications. Moreover, I have demonstrated the ability to do research independently in different sub-fields of Computer Science and succeeded in establishing highly fruitful collaborations. I also showed my potential as a group leader, in particular as the PI of the already mentioned WWTF project on argumentation. Thus, I am confident to have an excellent potential for succeeding with this challenging project.

My current position at TU Vienna runs until August 2013, thus the FWF START Project would be of great help at this ***critical stage of my career providing a final boost to establish myself as a fully independent researcher. Moreover, it should improve my position in negotiating for a permanent position at my university as well as in applications for professorship at other universities.***

6 Implications Beyond the Field

The methodology of the proposed research is of very general nature. In a nutshell, we aim at developing new methods for efficiently solving problems which involve *huge data* and *complex reasoning* over this data. As we have outlined, such problems provide two levels of structure both of them amenable to recent methods of Parameterized Complexity. It is obvious that this situation is not narrowed to the formalisms and query languages we focus within the project. Thus an impact on other branches of the field is predicted. Moreover, the particular languages we are investigating can be used in a wide range of applications. Since our methods are based on structure found in real-world data, we expect successful applications in other areas, in particular in Bio-Informatics.

A Bibliography

[1] A. Aavani, S. Tasharrofi, G. Ünel, E. Ternovska, and D. G. Mitchell. Speed-up techniques for negation in grounding. In *Proceedings LPAR-16*, volume 6355 of *LNCS*, pages 13–26. Springer, 2010.

[2] R. Agarwal, P. B. Godfrey, and S. Har-Peled. Approximate distance queries and compact routing in sparse graphs. In *Proceedings INFOCOM 2011*, pages 1754–1762. IEEE, 2011.

[3] H. Andréka, I. Németi, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

[4] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Alg. Disc. Meth.*, 8:277–284, 1987.

[5] A. Atserias, J. K. Fichte, and M. Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *J. Artif. Intell. Res. (JAIR)*, 40:353–373, 2011.

[6] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. CUP, 2003.

[7] M. Balduccini. Learning and using domain-specific heuristics in ASP solvers. *AI Commun.*, 24(2):147–164, 2011.

[8] M. Balduccini and T. C. Son, editors. *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning. Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday*, volume 6565 of *LNAI*. Springer, 2011.

[9] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. CUP, 2002.

[10] R. Barilaro, F. Ricca, and G. Terracina. Optimizing the distributed evaluation of stratified programs via structural analysis. In *Proceedings LPNMR 2011*, volume 6645 of *LNCS*, pages 217–222. Springer, 2011.

[11] B. Bliem, M. Morak, and S. Woltran. D-flat: Declarative problem solving using tree decompositions and answer-set programming. *TPLP*, 12(4-5):445–464, 2012.

[12] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[13] H. L. Bodlaender and A. M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008.

[14] H. L. Bodlaender and A. M. C. A. Koster. Treewidth computations I. Upper bounds. *Inf. Comput.*, 208(3):259–275, 2010.

[15] C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res. (JAIR)*, 21:135–191, 2004.

[16] G. Brewka, T. Eiter, and M. Truszczyński. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011.

[17] G. Brewka, M. Truszczynski, and S. Woltran. Representing preferences among sets. In *Proceedings AAAI 2010*, pages 273–278. AAAI Press, 2010.

[18] D. R. Brooks, E. Erdem, S. T. Erdogan, J. W. Minett, and D. Ringe. Inferring phylogenetic trees using answer set programming. *J. Autom. Reasoning*, 39(4):471–511, 2007.

[19] F. Buccafurri, N. Leone, and P. Rullo. Enhancing disjunctive datalog by constraints. *IEEE Trans. Knowl. Data Eng.*, 12(5):845–860, 2000.

[20] L. Cai, X. Huang, C. Liu, F. A. Rosamond, and Y. Song. Parameterized complexity and biopolymer sequence comparison. *Comput. J.*, 51(3):270–291, 2008.

[21] A. Calì, G. Gottlob, and T. Lukasiewicz. Datalog$^{\pm}$: a unified approach to ontologies and integrity constraints. In *Proceedings ICDT 2009*, pages 14–30. ACM, 2009.

[22] A. Calì, G. Gottlob, and T. Lukasiewicz. Tractable query answering over ontologies with datalog+/-. In *Proceedings DL 2009*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[23] A. Calì, G. Gottlob, and A. Pieris. New expressive languages for ontological query answering. In *Proceedings AAAI 2011*, pages 1541–1546. AAAI Press, 2011.

[24] F. Calimeri, S. Cozza, G. Ianni, and N. Leone. An ASP system with functions, lists, and sets. In *Proceedings LPNMR 2009*, volume 5753 of *LNCS*, pages 483–489. Springer, 2009.

[25] F. Calimeri, G. Ianni, F. Ricca, M. Alviano, A. Bria, G. Catalano, S. Cozza, W. Faber, O. Febbraro, N. Leone, M. Manna, A. Martello, C. Panetta, S. Perri, K. Reale, M. C. Santoro, M. Sirianni, G. Terracina, and P. Veltri. The third answer set programming competition: Preliminary report of the system competition track. In *Proceedings LPNMR 2011*, volume 6645 of *LNCS*, pages 388–403. Springer, 2011.

[26] B. Courcelle. Graph rewriting: An algebraic and logic approach. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 193–242. The MIT Press, 1990.

[27] C. Cumbo, W. Faber, G. Greco, and N. Leone. Enhancing the magic-set method for disjunctive datalog programs. In *Proceedings ICLP 2004*, volume 3132 of *LNCS*, pages 371–385. Springer, 2004.

[28] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[29] M. Dao-Tran, T. Eiter, M. Fink, and T. Krennwallner. Modular nonmonotonic logic programming revisited. In *Proceedings ICLP 2009*, volume 5649 of *LNCS*, pages 145–159. Springer, 2009.

[30] A. Dermaku, T. Ganzow, G. Gottlob, B. J. McMahan, N. Musliu, and M. Samer. Heuristic methods for hypertree decomposition. In *Proceedings MICAI 2008*, volume 5317 of *LNCS*, pages 1–11. Springer, 2008.

[31] G. Dong. On distributed processibility of datalog queries by decomposing databases. In *Proceedings SIGMOD 1989*, pages 26–35. ACM Press, 1989.

[32] F. Dorn and J. A. Telle. Semi-nice tree-decompositions: The best of branchwidth, treewidth and pathwidth with one algorithm. *Discrete Applied Mathematics*, 157(12):2737–2746, 2009.

[33] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

[34] R. G. Downey, M. R. Fellows, and M. A. Langston. The computer journal special issue on parameterized complexity: Foreword by the guest editors. *Comput. J.*, 51(1):1–6, 2008.

[35] W. Dvorák, R. Pichler, and S. Woltran. Towards fixed-parameter tractable algorithms for abstract argumentation. *Artif. Intell.*, 186:1–37, 2012.

[36] W. Dvořák, R. Pichler, and S. Woltran. Towards fixed-parameter tractable algorithms for argumentation. In *Proceedings KR 2010*, pages 112–122. AAAI Press, 2010.

[37] W. Dvořák, S. A. Gaggl, J. P. Wallner, and S. Woltran. Making use of advances in answer-set programming for abstract argumentation systems. *CoRR*, abs/1108.4942, 2011.

[38] W. Dvořák, S. Szeider, and S. Woltran. Reasoning in argumentation frameworks of bounded clique-width. In *Proceedings COMMA'10*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 219–230. IOS Press, 2010.

[39] U. Egly, S. A. Gaggl, and S. Woltran. Answer-set programming encodings for argumentation frameworks. *Argument & Computation*, 1(2):147–177, 2010.

[40] T. Eiter, W. Faber, M. Fink, and S. Woltran. Complexity results for answer set programming with bounded predicate arities and implications. *Ann. Math. Artif. Intell.*, 51(2-4):123–165, 2007.

[41] T. Eiter, W. Faber, and M. Mushthofa. Space efficient evaluation of ASP programs with bounded predicate arities. In *Proceedings AAAI 2010*, pages 303–308, 2010.

[42] T. Eiter, M. Fink, H. Tompits, P. Traxler, and S. Woltran. Replacements in non-ground answer-set programming. In *Proceedings KR 2006*, pages 340–351. AAAI Press, 2006.

[43] T. Eiter, M. Fink, H. Tompits, and S. Woltran. Strong and uniform equivalence in answer-set programming: Characterizations and complexity results for the non-ground case. In *Proceedings AAAI 2005*, pages 695–700. AAAI Press, 2005.

[44] T. Eiter, M. Fink, and S. Woltran. Semantical characterizations and complexity of equivalences in answer set programming. *ACM Trans. Comput. Log.*, 8(3), 2007.

[45] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive datalog. *ACM Trans. Database Syst.*, 22(3):364–418, 1997.

[46] T. Eiter and M. Simkus. FDNC: Decidable nonmonotonic disjunctive logic programs with function symbols. *ACM Trans. Comput. Log.*, 11(2), 2010.

[47] E. Erdem, Y. Erdem, H. Erdogan, and U. Öztok. Finding answers and generating explanations for complex biomedical queries. In *Proceedings AAAI 2011*, pages 785–790. AAAI Press, 2011.

[48] W. Faber, G. Pfeifer, and N. Leone. Semantics and complexity of recursive aggregates in answer set programming. *Artif. Intell.*, 175(1):278–298, 2011.

[49] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[50] T. Fayruzov, J. Janssen, D. Vermeir, C. Cornelis, and M. D. Cock. Modelling gene and protein regulatory networks with answer set programming. *IJDMB*, 5(2):209–229, 2011.

[51] M. R. Fellows. Parameterized complexity: The main ideas and connections to practical computing. In *Proceedings ISAAC 2001*, volume 2223 of *LNCS*, pages 291–307. Springer, 2001.

[52] J. K. Fichte and S. Szeider. Backdoors to tractable answer-set programming. In *Proceedings IJCAI 2011*, pages 863–868. IJCAI/AAAI, 2011.

[53] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[54] M. Gebser, T. Grote, R. Kaminski, and T. Schaub. Reactive answer set programming. In *Proceedings LPNMR 2011*, volume 6645 of *LNCS*, pages 54–66. Springer, 2011.

[55] M. Gebser, C. Guziolowski, M. Ivanchev, T. Schaub, A. Siegel, S. Thiele, and P. Veber. Repair and prediction (under inconsistency) in large biological networks with answer set programming. In *Proceedings KR*, pages 497–507. AAAI Press, 2010.

[56] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. Challenges in answer-set programming. In *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday*, volume 6565 of *LNAI*, pages 74–90. Springer, 2011.

[57] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, M. T. Schneider, and S. Ziller. A portfolio solver for answer set programming: Preliminary report. In *Proceedings LPNMR 2011*, volume 6645 of *LNCS*, pages 352–357. Springer, 2011.

[58] M. Gebser, R. Kaminski, M. Knecht, and T. Schaub. plasp: A prototype for PDDL-based planning in ASP. In *Proceedings LPNMR 2011*, volume 6645 of *LNCS*, pages 358–363. Springer, 2011.

[59] M. Gebser, R. Kaminski, and T. Schaub. Complex optimization in answer set programming. *TPLP*, 11(4-5):821–839, 2011.

[60] M. Gebser, B. Kaufmann, R. Kaminski, M. Ostrowski, T. Schaub, and M. T. Schneider. Potassco: The Potsdam answer set solving collection. *AI Commun.*, 24(2):107–124, 2011.

[61] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. Advanced preprocessing for answer set solving. In *Proceedings ECAI 2008*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 15–19. IOS Press, 2008.

[62] M. Gebser, B. Kaufmann, and T. Schaub. The conflict-driven answer set solver clasp: Progress report. In *Proceedings LPNMR 2009*, volume 5753 of *LNCS*, pages 509–514. Springer, 2009.

[63] M. Gebser, O. Sabuncu, and T. Schaub. An incremental answer set programming based system for finite model computation. In *Proceedings JELIA 2010*, volume 6341 of *LNCS*, pages 169–181. Springer, 2010.

[64] M. Gebser, T. Schaub, and S. Thiele. Gringo : A new grounder for answer set programming. In *Proceedings LPNMR 2007*, volume 4483 of *LNCS*, pages 266–271. Springer, 2007.

[65] M. Gebser, T. Schaub, S. Thiele, and P. Veber. Detecting inconsistencies in large biological networks with answer set programming. *TPLP*, 11(2-3):323–360, 2011.

[66] A. Gelfand, K. Kask, and R. Dechter. Stopping rules for randomized greedy triangulation schemes. In *Proceedings AAAI 2011*, pages 1043–1048. AAAI Press, 2011.

[67] M. Gelfond. Representing knowledge in A-Prolog. In *Computational Logic: From Logic Programming into the Future*, volume 2408 of *LNCS/LNAI*, pages 413–451. Springer, 2002.

[68] M. Gelfond and N. Leone. Logic programming and knowledge representation - the A-Prolog perspective. *Artif. Intell.*, 138(1-2):3–38, 2002.

[69] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4):365–386, 1991.

[70] L. Ghionna, L. Granata, G. Greco, and F. Scarcello. Hypertree decompositions for query optimization. In *Proceedings ICDE 2007*, pages 36–45. IEEE, 2007.

[71] E. Giunchiglia, N. Leone, and M. Maratea. On the relation among answer set solvers. *Ann. Math. Artif. Intell.*, 53(1-4):169–204, 2008.

[72] V. Gogate and R. Dechter. A complete anytime algorithm for treewidth. In *Proceedings UAI 2004*, pages 201–208. AUAI Press, 2004.

[73] J. Goldsmith, J. Lang, M. Truszczynski, and N. Wilson. The computational complexity of dominance and consistency in CP-nets. *J. Artif. Intell. Res. (JAIR)*, 33:403–432, 2008.

[74] G. Gottlob, E. Grädel, and H. Veith. Datalog LITE: a deductive query language with linear time model checking. *ACM Trans. Comput. Log.*, 3(1):42–79, 2002.

[75] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. *J. Comput. Syst. Sci.*, 64(3):579–627, 2002.

[76] G. Gottlob, R. Pichler, and F. Wei. Bounded treewidth as a key to tractability of knowledge representation and reasoning. *Artif. Intell.*, 174(1):105–132, 2010.

[77] G. Gottlob, R. Pichler, and F. Wei. Monadic datalog over finite structures of bounded treewidth. *ACM Trans. Comput. Log.*, 12(1), 2010.

[78] G. Gottlob, F. Scarcello, and M. Sideri. Fixed-parameter complexity in AI and nonmonotonic reasoning. *Artif. Intell.*, 138(1-2):55–86, 2002.

[79] E. Grädel. Invited talk: Decision procedures for guarded logics. In *Proceedings CADE-16*, volume 1632 of *LNCS*, pages 31–51. Springer, 1999.

[80] J. Gramm, A. Nickelsen, and T. Tantau. Fixed-parameter algorithms in phylogenetics. *Comput. J.*, 51(1):79–101, 2008.

[81] M. A. Hall and L. A. Smith. Practical feature subset selection for machine learning. In *Proceedings ACSC 1998*, pages 181–191. Springer, 1998.

[82] S. Heymans, D. V. Nieuwenborgh, and D. Vermeir. Open answer set programming with guarded programs. *ACM Trans. Comput. Log.*, 9(4), 2008.

[83] P. Hlinený, S. Oum, D. Seese, and G. Gottlob. Width parameters beyond tree-width and their applications. *The Computer Journal*, 51(3):326–362, 2008.

[84] X. Huang and J. Lai. Parameterized graph problems in computational biology. In *Proceedings IMSCCS 2007*, pages 129–132. IEEE, 2007.

[85] Z. Huang, Y. Wu, J. Robertson, L. Feng, R. L. Malmberg, and L. Cai. Fast and accurate search for non-coding RNA pseudoknot structures in genomes. *Bioinformatics*, 24(20):2281–2287, 2008.

[86] U. Hustadt, B. Motik, and U. Sattler. Reasoning in description logics by a reduction to disjunctive datalog. *J. Autom. Reasoning*, 39(3):351–384, 2007.

[87] G. Ianni, A. Martello, C. Panetta, and G. Terracina. Efficiently querying RDF(S) ontologies with answer set programming. *J. Log. Comput.*, 19(4):671–695, 2009.

[88] M. Jakl, R. Pichler, and S. Woltran. Answer-set programming with bounded treewidth. In *Proceedings IJCAI 2009*, pages 816–822. AAAI Press, 2009.

[89] K. Kask, A. Gelfand, L. Otten, and R. Dechter. Pushing the power of stochastic greedy ordering schemes for inference in graphical models. In *Proceedings AAAI 2011*, pages 54–60. AAAI Press, 2011.

[90] P. G. Kolaitis, J. Panttaja, and W. C. Tan. The complexity of data exchange. In *Proceedings PODS 2006*, pages 30–39. ACM, 2006.

[91] A. Langer, F. Reidl, P. Rossmanith, and S. Sikdar. Evaluation of an MSO-solver. In *Proceedings ALENEX 2012*, pages 55–63. SIAM / Omnipress, 2012.

[92] M. Latapy and C. Magnien. Measuring fundamental properties of real-world complex networks. *CoRR*, abs/cs/0609115, 2006.

[93] C. Lefèvre and P. Nicolas. A first order forward chaining approach for answer set computing. In *Proceedings LPNMR 2009*, volume 5753 of *LNCS*, pages 196–208. Springer, 2009.

[94] M. Lenzerini. Data integration: A theoretical perspective. In *Proceedings PODS 2002*, pages 233–246. ACM, 2002.

[95] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Log.*, 7(3):499–562, 2006.

[96] V. Lifschitz. What is answer set programming? In *Proceedings AAAI 2008*, pages 1594–1597. AAAI Press, 2008.

[97] V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Trans. Comput. Log.*, 2(4):526–541, 2001.

[98] F. Lin and X. Zhao. On odd and even cycles in normal logic programs. In *Proceedings AAAI 2004*, pages 80–85. AAAI Press / The MIT Press, 2004.

[99] M. Maratea, L. Pulina, and F. Ricca. The multi-engine ASP solver me-asp. In *Proceedings JELIA*, volume 7519 of *LNCS*, pages 484–487. Springer, 2012.

[100] V. W. Marek and M. Truszczyński. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm – A 25-Year Perspective*, pages 375–398. Springer, 1999.

[101] D. L. McGuinness and J. R. Wright. An industrial strength description logic-based configuration platform. *IEEE Intelligent Systems*, 13(4):69–77, 1998.

[102] G. Melançon. Just how dense are dense graphs in the real world? A methodological note. In *Proceedings BELIV 2006*, pages 1–7. ACM Press, 2006.

[103] V. S. Mellarkod, M. Gelfond, and Y. Zhang. Integrating answer set programming and constraint logic programming. *Ann. Math. Artif. Intell.*, 53(1-4):251–287, 2008.

[104] M. Morak, N. Musliu, R. Pichler, S. Rümmele, and S. Woltran. Evaluating tree-decomposition based algorithms for answer set programming. Technical Report DBAI-TR-2011-73, Vienna Univ. of Technology, 2011.

[105] M. Morak, N. Musliu, R. Pichler, S. Rümmele, and S. Woltran. A new tree-decomposition based algorithm for answer set programming. In *Proceedings ICTAI*, pages 916–918. IEEE, 2011.

[106] M. Morak, R. Pichler, S. Rümmele, and S. Woltran. A dynamic-programming based ASP-solver. In *Proceedings JELIA 2010*, volume 6341 of *LNCS*, pages 369–372. Springer, 2010.

[107] M. Morak and S. Woltran. Preprocessing of complex non-ground rules in answer set programming. In *Proceedings ICLP*, volume 17 of *LIPIcs*, pages 247–258. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[108] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. OUP, 2006.

[109] I. Niemelä. Logic programming with stable model semantics as a constraint programming paradigm. *Ann. Math. Artif. Intell.*, 25(3–4):241–273, 1999.

[110] I. Niemelä. Stable models and difference logic. *Ann. Math. Artif. Intell.*, 53(1-4):313–329, 2008.

[111] M. Nogueira, M. Balduccini, M. Gelfond, R. Watson, and M. Barry. An A-Prolog decision support system for the Space Shuttle. In *Proceedings PADL 2001*, volume 1990 of *LNCS*, pages 169–183. Springer, 2001.

[112] A. D. Palù, A. Dovier, E. Pontelli, and G. Rossi. GASP: Answer set programming with lazy grounding. *Fundam. Inform.*, 96(3):297–322, 2009.

[113] P. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language semantics and abstract syntax – W3C recommendation. Technical report, World Wide Web Consortium, Feb. 2004. Available at `http://www.w3.org/TR/owl-semantics/`.

[114] R. Pichler, S. Rümmele, S. Szeider, and S. Woltran. Tractable answer-set programming with weight constraints: Bounded treewidth is not enough. In *Proceedings KR 2010*, pages 508–517. AAAI Press, 2010.

[115] R. Pichler, S. Rümmele, and S. Woltran. Belief revision with bounded treewidth. In *Proceedings LPNMR 2009*, volume 5753 of *LNCS*, pages 250–263. Springer, 2009.

[116] R. Pichler, S. Rümmele, and S. Woltran. Multicut algorithms via tree decompositions. In *Proceedings CIAC 2010*, volume 6078 of *LNCS*, pages 167–179. Springer, 2010.

[117] A. Poggi, D. Lembo, D. Calvanese, G. D. Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

[118] N. Robertson and P. D. Seymour. Graph minors II: Algorithmic aspects of tree-width. *Journal Algorithms*, 7:309–322, 1986.

[119] R. Ronen and O. Shmueli. Evaluating very large datalog queries on social networks. In *Proceedings EDBT 2009*, pages 577–587. ACM, 2009.

[120] M. Samer and S. Szeider. Algorithms for propositional model counting. *J. Discrete Algorithms*, 8(1):50–64, 2010.

[121] F. Scarcello, G. Greco, and N. Leone. Weighted hypertree decompositions and optimal query plans. *J. Comput. Syst. Sci.*, 73(3):475–506, 2007.

[122] J. Shao, D. A. Bell, and M. E. C. Hull. Combining rule decomposition and data partitioning in parallel datalog program processing. In *Proceedings PDIS 1991*, pages 106–115. IEEE, 1991.

[123] P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artif. Intell.*, 138(1-2):181–234, 2002.

[124] K. Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Comput. Surv.*, 41(1), 2008.

[125] K. T. Tekle and Y. A. Liu. More efficient datalog queries: subsumptive tabling beats magic sets. In *Proceedings SIGMOD 2011*, pages 661–672. ACM, 2011.

[126] M. Thorup. All structured programs have small tree-width and good register allocation. *Information and Computation*, 142(2):159–181, 1998.

[127] M. Truszczynski. Computing large and small stable models. *TPLP*, 2(1):1–23, 2002.

[128] J. M. M. van Rooij, H. L. Bodlaender, and P. Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In *Proceedings ESA 2009*, volume 5757 of *LNCS*, pages 566–577. Springer, 2009.

[129] M. Vardi. On the complexity of bounded-variable queries. In *Proceedings PODS 1995*, pages 266–276. ACM Press, 1995.

[130] J. Xu, F. Jiao, and B. Berger. A tree-decomposition approach to protein structure prediction. In *Proceedings IEEE Computational Systems Bioinformatics Conference*, pages 247–256, 2005.

[131] L. Xu, H. Hoos, and K. Leyton-Brown. Hydra: Automatically configuring algorithms for portfolio-based selection. In *Proceedings AAAI 2010*, pages 210–216. AAAI Press, 2010.

[132] A. Yamaguchi, K. F. Aoki, and H. Mamitsuka. Graph complexity of chemical compounds in biological pathways. *Genome Informatics*, 14:376–377, 2003.

[133] Y. Zabiyaka and A. Darwiche. Functional treewidth: Bounding complexity in the presence of functional dependencies. In *Proceedings SAT 2006*, volume 4121 of *LNCS*, pages 116–129. Springer, 2006.

[134] N.-F. Zhou, Y. Kameya, and T. Sato. Mode-directed tabling for dynamic programming, machine learning, and constraint solving. In *Proceedings ICTAI 2010*, pages 213–218. IEEE, 2010.

[135] N.-F. Zhou, T. Sato, and Y.-D. Shen. Linear tabling strategies and optimizations. *TPLP*, 8(1):81–109, 2008.