WCB 12

Workshop on Constraint-Based Methods for Bioinformatics

September 8^{th}, 2012

Proceedings

Rolf Backofen and Sebastian Will

Preface

This volume contains the contributions to WCB12, the eighth Workshop on Constraint Based Methods for Bioinformatics. The meeting was held on September 8, 2012 in Budapest colocated with the International Conference on Logic Programming (ICLP12). For these proceedings and presentation at the meeting, the committee decided to accept all of the 5 strong submissions after thorough peer-review. Each paper was reviewed by 3-4 program committee members.

The workshop series has its origins already 14 years ago with the workshops "Constraints and Bioinformatics/Biocomputing". These workshops have been colocated with the constraint programming conferences CP97 and CP98. Starting from 2005, the Workshop on Constraint based methods for Bioinformatics (WCB) series took place every year, colocated with either constraint or logic programming conferences. The numerous submissions to the workshops over the past years demonstrate impressively that constraint-based techniques provide solutions for various complex bioinformatics tasks, comprising problems from sequence alignment, simulation of biological systems, haplotype inference, pedigree analysis, structure prediction of RNAs and proteins, and protein docking. For this edition of WCB, we collected submissions discussing (in random order) loop modeling for protein structure prediction, modeling of cell dynamics, analysis of biochemical networks, mapping of atoms in chemical reactions, and a web service providing phylogenetic data.

We thank our colleagues Nicos Angelopoulos, Pedro Barahona, Alexander Bockmayer, Alessandro Dal Palu, Agostino Dovier, Esra Erdem, Francois Fages, Andrea Formisano, Ines Lynce, Enrico Ponticelli, Sven Thiele, and Pascal van Hentenryck for supporting the workshop by joining the program committe. Moreover, we thank all authors who contributed their papers to WCB12. Our special thanks go to our invited speaker Christoph Flamm for enhancing the meeting with his keynote talk. Finally, we thank the organizers of ICLP12 and, in particular, the workshop chair Mats Carlsson for setting the comfortable friendly environment that made this workshop possible. Finally, we thank the developers of EasyChair; their system greatly supported the reviewing process and generation of the proceedings.

August 1st, 2012 — Sebastian Will
Freiburg — Rolf Backofen

Table of Contents

Program Committee

Nicos Angelopoulos	Netherlands Cancer Institute
Rolf Backofen	Albert-Ludwigs-University Freiburg
Pedro Barahona	Universidade Nova de Lisboa
Alexander Bockmayer	Freie Universität Berlin
Alessandro Dal Palu	Università degli Studi di Parma
Agostino Dovier	Univ. di UDINE
Esra Erdem	Sabanci University
Francois Fages	INRIA Rocquencourt
Andrea Formisano	Dip. di Matematica e Informatica, Univ. di Perugia
Ines Lynce	IST/INESC-ID, Technical University of Lisbon
Enrico Ponticelli	New Mexico State University
Sven Thiele	University of Potsdam
Pascal van Hentenryck	Brown University
Sebastian Will	Albert-Ludwigs-Universitaet Freiburg

Search and Rescue: logic and visualisation of biochemical networks

Nicos Angelopoulos[1], Paul Shannon[2], and Lodewyk Wessels[1]

[1] Netherlands Cancer Institute, Amsterdam, Netherlands
[2] Fred Hutchinson Cancer Research Center, Seattle, USA
n.angelopoulos@nki.nl

Abstract. We utilise a recently introduced Prolog package that allows communication with the *R* statistical software to develop a graph-centric suit of procedures that allow the exploration of biological data and hit-lists from within Prolog. We show how a number of public protein-protein interaction databases can be intuitively be represented as facts before we present search algorithms that are naturally expressed in logical terms. Visualisation of the resulting graphs is done via low-level communication to *Bioconductor's RCytoscape* package. We illustrate the utility of the integrative Prolog platform using two public databases and a graph search to connect elements of genes involved in cell motility.

1 Introduction

Constraint logic programming is a powerful yet under-valued platform for research and analysis in bioinformatics and computational biology. Scripting, high-level of abstraction, interpreter-base and automatic memory management are all features of logic programming (LP) that make it ideal for the development of research-led code in the above areas.

The areas in which LP is deficient are: the lack of user-based package extensions as typified by code repositories, the limited number of statistical packages and its graphical abilities. The deployment of LP in areas that have a strong ethos with regard to backing conceptual ideas and research results with functioning code will help create the necessary conditions for code repositories. This has already been born out for the communities of *R* [7], *Perl* and *Python*. Where, their use in bioinformatics has substantially bolstered their community-contributed code base. One might argue, that a surge in the use of LP in computational biology would be beneficial for its expansion. This is seriously hampered by the second and third shortcomings, i.e. the lack of statistical reasoning and graphical output software suits. The use of LP has been previously argued and used in bioinformatics and particular in ontology reasoning [6].

In this paper we take advantage of recent developments in integrating *R* within Prolog [1] to explore graph searching and visualisation within logic programming. The strengths of Prolog in data representation and search are put into representing and reasoning with biological knowledge. Furthermore, the complementary strengths of *R* in visualisation are brought to bear within a logic programming environment, thus doing away with need of re-implementing such procedures.

The remainder of the paper is organised as follows. Section 2 describes protein-protein interactions networks (PPIs). Section 3 presents graph operations on gene lists in the context of PPIs and develops these ideas on a specific PPI and motility gene list. The paper's concluding remarks are in the Section 4.

2 Protein-Protein interaction networks

The last decades have witnessed a phenomenal increase in the amount of biological knowledge that has been published and codified. This acceleration can be directly attributed to the evolution of high throughput technologies such as genome wide expression assays, microscopy, and deep sequencing.

One important way in which biological knowledge is codified is in the form of protein-protein interaction (PPI) databases such as STRING [9] and HPRD [4]. STRING collates information from a variety of sources including predicted interactions and gives weight scores to each edge based on the strength of the evidence supporting the corresponding interaction. It currently contains information on $5,214,234$ proteins from 1133 organisms and holds $224,346,017$ interactions. HPRD holds human proteins and interactions between them. Currently there are $39,194$ interactions in HPRD.

Apart from direct interactions it is also straight forward to represent interactions passing through metabolites, such as the interactions present in the metabolic pathways present in the KEGG database [3]. Visualising PPIs are often in the form of networks/graphs which provide an overall picture of the connectivity between the various pathways mapping biological functions.

Representing these types of interactions depends to a large extent on the operations one plans to perform. One way in which directional HPRD interactions can be stored is as interaction facts:

```
interaction( From, To, Types, References ).
```

Proteins *From* and *To* are mapped to Entrez IDs and *Types* are the types of evidence provided in *References*. Alternatively, when type of interaction needs to be explicitly represented, as is the case in our KEGG database example below, one can represent the separate reaction types as separate facts:

```
activation( From, To, Organism, Pathway ).
inhibition( From, To, Organism, Pathway ).
phosphorylation( From, To, Organism, Pathway ).
ubiquination( From, To, Organism, Pathway ).
```

The represented interaction of the involved proteins is known to occur in the specified *Organism* and is part of the KEGG pathway identified by $Pathway$.

As regulatory, metabolic and signalling networks become better known, due largely to advances in laboratory technique, we will nonetheless be faced by the condition-specific, and cell-line specific nature of all molecular interactions within the cell. Logic programming offers new capabilities to understand these interactions, and in our efforts to predict and control cellular behaviour.

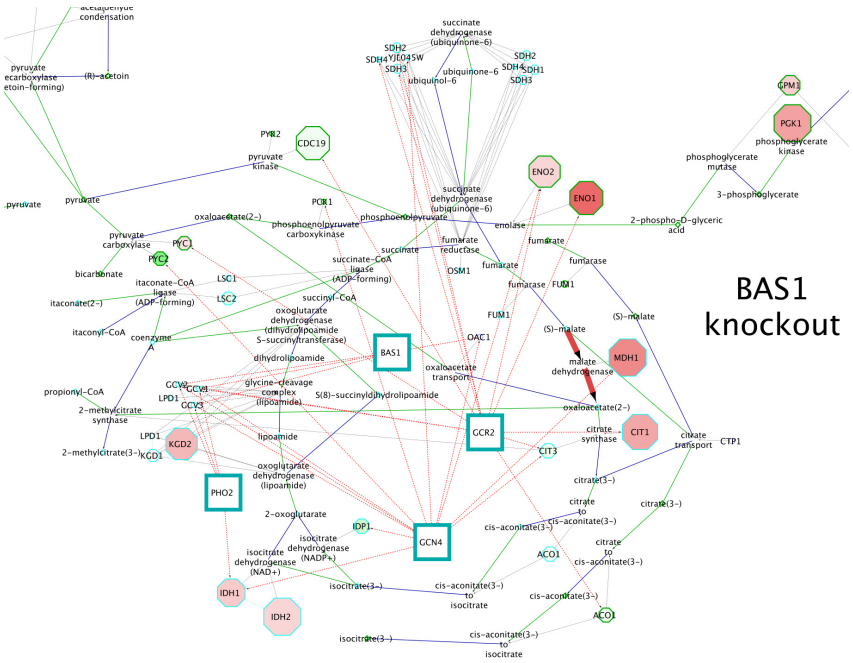

Fig. 1. Transcription factors control metabolic pathways activity in yeast [2].

The complexity and contingency of regulatory networks is an emerging, recurrent theme demonstrated in a plethora of computational biology papers. For instance, [2] describes a combination of metabolite flux measurements and protein abundance across 119 transcription factor knockout strains in yeast, to identify a small number of transcription factors which regulate a crucial step in the TCA cycle. Using *RCytoscape*, we displayed these on top of the yeast consensus carbon cycle metabolic network (`http://www.comp-sys-bio.org/yeastnet/`).

Logic programming is a natural complement to this network visualisation of the data in Fendt et. al [2]- which is multi-dimensional, and representative of the rich networks which will become increasingly available. Representing and reasoning with the multiple levels of such networks that include regulatory, metabolic and signalling components can be an important future research area for logic programming. Its AI heritage can be put in good use in elucidating the intricate details of such structures and inferentially associate or predict the outcome of interventions.

3 Gene-lists in Graphs

A variety of bioinformatics analyses have as an end or intermediate product the generation of a list of genes. Visualising these lists in the context of protein interaction

Fig. 2. KEGG interactions for a subset of the members of an adhesome library. Nodes are proteins and edges denote interactions. Blue nodes are connecting proteins that do not appear in the library. Edges are coloured as per type of interaction.

networks is a useful tool through which results can be presented. Visualisation is particularly powerful when communicating results to experimental biologists.

Cell motility is a complex biological process that plays a critical role in development, wound healing and cancer metastasis. It is mediated by focal adhesion complexes which are dynamic structures that may involve a large number of proteins. There exists a large body of literature that studies the molecular mechanisms by which cells move in-vitro and in-vivo. The main core of the proteins involved in complexes has been placed by some studies to 156 [11] while when considering potential encoding genes for the whole motility apparatus, the total number can be substantially greater. Such broadly defined libraries play an important role in screening programmes. Here we explore a set of 570 motility related genes that were gathered from a variety of sources [10].

3.1 Graph operations

We mapped the list of motility genes to the KEGG interaction pathways. We construct a graph by adding an edge between any pair of genes with a known interaction in KEGG. As only a limited number of pairs have direct interactions we extended the graph by implementing a depth first search algorithm based on the representation of KEGG interactions we already discussed. The most connected sub-graph is designated as the seed of the main graph and attempts are made to expand it. For each of the remaining sub-graphs or disconnected motility genes, a breadth first attempt is made to connect

it to the main graph by adding n additional nodes. If n such nodes can be found, the current sub-graph is removed from the list of sub-graphs to be connected, otherwise the algorithm repeats the test for $n = n + 1$. The algorithm is greedy in that it only analyzes the first extending path of n additional genes it encounters. The sub-graph is removed from the list of sub-graph to be connected if there exists no n such that it can be connected to the main graph. The algorithm terminates when it encounters an empty sub-graphs list. The implementation of the algorithm in Prolog is elegant and easy to communicate and maintain. We hope it will be added to the standard graph operations library that exists in many Prolog systems.

3.2 Visualisation software

The results of the algorithm on the motility list are shown in Figure 2. Note that not all genes can be connected in this case. Some of these disconnected genes are shown at the bottom of the graph. Visualisation has been one of the areas in which Prolog has been weak. Here we utilise *r.eal* [1], a recently developed Prolog library that allows efficient interactions with the *R* statistical software system. We developed software that allows the visualisation of Prolog graphs via the *RCytoscape Bioconductor* package [8]. Our interface code can be used with Prolog represented graphs from the standard graph library and provides convenient options for rendering a variety of aspects for all graph elements. It is worth noting that the interaction between Prolog and *Cytoscape* is bi-directional. Sets of nodes selected via the graphical interface can be interactively accessed via Prolog.

4 Conclusions

We have argued in this paper that Prolog is a power platform for data analysis and computational research in biofinformatics. Biological knowledge can be succinctly represented and reasoned about within logic programming which has traditional strengths in artificial intelligence research and provides a high-level at which one can interact with biological datasets.

In addition, we present practical steps towards the promotion of logic programming in the manipulation and visualisation of biochemical networks and associated gene lists. Graph operation on such lists are crucial to communicating results of analysis to experimentalists but can also provide the basis for further analysis. For example, in network based regression algorithms [5]. Our software is a useful addition to existing Prolog code in the bioinformatics domain [6].

References

1. Angelopoulos, N., Costa, V.S., Azevedo, J., Camacho, R., Wessels, L.: Integrative statistics for logical reasoning. In preparation, `http://bioinformatics.nki.nl/~nicos/sware/real` (2012)
2. Fendt, S.M., Oliveira, A.P., Christen, S., Picotti, P., Dechant, R.C., Sauer, U.: Unraveling condition-dependent networks of transcription factors that control metabolic pathway activity in yeast. Mol Syst Biol **6** (2010)

3. Kanehisa, M., Goto, S., Sato, Y., Furumichi, M., Tanabe, M.: Kegg for integration and interpretation of large-scale molecular datasets. Nucleic Acids Res. (2012) D109–D114
4. Keshava Prasad, T.S., Goel, R., Kandasamy, K., Keerthikumar, S., Kumar, S., Mathivanan, S., Telikicherla, D., Raju, R., Shafreen, B., Venugopal, A., Balakrishnan, L., Marimuthu, A., Banerjee, S., Somanathan, D.S., Sebastian, A., Rani, S., Ray, S., Harrys Kishore, C.J., Kanth, S., Ahmed, M., Kashyap, M.K., Mohmood, R., Ramachandra, Y.L., Krishna, V., Rahiman, B.A., Mohan, S., Ranganathan, P., Ramabadran, S., Chaerkady, R., Pandey, A.: Human protein reference database2009 update. Nucleic Acids Research **37**(suppl 1) (2009) D767–D772
5. Maathuis, M.H., Colombo, D., Kalisch, M., Bhlmann, P.: Predicting causal effect in large-scale systems from observational data. Nature Methods **7**(4) (2010) 247–248
6. Mungall, C.: Experiences using logic programming in bioinformatics. In Hill, P., Warren, D., eds.: Logic Programming, 25th International Conference, ICLP 2009. Volume 5649 of Lecture Notes in Computer Science. Springer (2009) 1–21
7. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. (2011)
8. Shannon, P.: RCytoscape: Display and manipulate graphs in Cytoscape. (2012) R package version 1.6.3.
9. Szklarczyk, D., Franceschini, A., Kuhn, M., Simonovic, M., Roth, A., Minguez, P., Doerks, T., Stark, M., Muller, J., Bork, P., Jensen, L.J., Mering, C.v.: The string database in 2011: functional interaction networks of proteins, globally integrated and scored. Nucleic Acids Research **39**(suppl 1) (2011) D561–D568
10. van Roosmalen, W.: Motility involved genes (2012) Personal communication.
11. Zaidel-Bar, R., Itzkovitz, S., Ma'ayan, A., Iyengar, R., Geiger, B.: Functional atlas of the integrin adhesome. Nature Cell Biology **9**(8) (08 2007) 858–867

Protein Loop Modeling via Constraints and Fragment Assembly

F. Campeotto[1,2], A. Dal Palù[3], A. Dovier[2], F. Fioretto[1], and E. Pontelli[1]

[1] Dept. Computer Science, New Mexico State University
[2] Depts. Math. & Computer Science, Univ. Udine
[3] Dept. Mathematics, Univ. Parma

Abstract. Methods to predict the structure of a protein often rely on the knowledge of macro-sub-structures and their exact or approximate relative positions in space. The parts connecting these sub-structures are called *loops* and, in general, they are characterized by a high degree of freedom. Modeling proteins loops is thus a critical problem in predicting protein conformations that are biologically realistic. This paper introduces a novel constraint targeted at modeling proteins loops with fragment assembly, and presents a filtering technique, inspired by inverse kinematics, that can drastically reduce the search space of potential conformations.

1 Introduction

Proteins are macro-molecules of fundamental importance in the way they regulate vital functions in all biological processes. These function are in a direct correspondence with the protein's 3D structure and, due to its inherent and computational complexity [1, 5], investigating its spatial conformation is one of the most important open problems in the bioinformatics field. This has originated a variety of alternative approaches. One method, named *fragments assembly*, has proved to be particularly promising. The idea behind this method is to assemble a protein structure by using small protein subunits as templates that present similarities (*homologous affinity*) w.r.t. the object sequence.

Nevertheless, even when protein structure prediction is realized using homologous templates, the final conformation may present aperiodic structures (*loops*) connecting the known protein segments on the outer region of the protein. These protein regions are, in general, not conserved during evolution, and therefore templates provide very limited statistical structural information. Modeling a protein loop often imposes constraints in the way of connecting two protein segments. Restrictions on the mutual positions and orientations (dihedral angles) of the loop anchors are often present. Such restrictions are defined as the *loop closure* constraints (Fig. 1). Popular methods for loop prediction include the *CCD* [3] and the *SOS* algorithm [7].

In this paper, we adopt *Constraint Programming (CP)* techniques to encode such constraints and, together with *fragments assembly*, we investigate the problem of *protein loop modeling*. In particular, we abstract the problem as a general multi-body system, where each composing body is constrained by means of geometric properties in the space and related to other bodies through joint relationships. This model leads to the *Joined-Multibody* (JM) constraint. Realistic loop modeling requires the assembly of hundreds of different body versions, making the problem intractable. We study an efficient approximated propagator, named *JM filtering (JMf)*, whose propagation is performed by an ad-hoc algorithm. This propagator allows us to efficiently compute classes of solutions, partitioned by structural similarity and controlled tolerance.

We demonstrate the strength of the filtering algorithm in significantly reducing the search space and in aiding the selection of representative solutions.

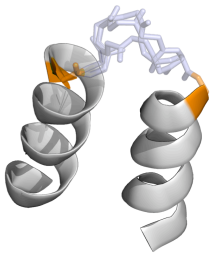

Fig. 1: Helices with a loop. The loop anchors are colored in orange; the loop constraint is satisfied by the loops connecting the two anchor points.

2 The Joined-Multibody Constraint

A *rigid block* B is composed of an ordered list of at least three 3D points, denoted by $\mathsf{points}(B)$, represented by circles in Figure 2. The *anchors* and *end-effectors* of a rigid block B, denoted by $\mathsf{start}(B)$ and $\mathsf{end}(B)$, are the two lists containing the first three and the last three points of $\mathsf{points}(B)$. With $B(i)$ we denote the i-th point of the rigid block B. For two ordered lists of points $\boldsymbol{p}$ and $\boldsymbol{q}$, we write $\boldsymbol{p} \frown \boldsymbol{q}$ if they can be perfectly overlapped by a rigid translation and/or rotation (i.e., a roto-translation).

Definition 1 (Multi-body). *A sequence $S_1, \ldots, S_n$ of non-empty sets of rigid blocks is said to be a* multi-body*. A sequence of rigid blocks $B_1, \ldots, B_n$, is called a* rigid body *if, for all $i = 1, \ldots, n-1$, $\mathsf{end}(B_i) \frown \mathsf{start}(B_{i+1})$.*

A rigid body can be seen as one instance of a multi-body that guarantees the partial overlapping of each two consecutive blocks. The overlapped points $\mathsf{end}(B_i)$ and $\mathsf{start}(B_{i+1})$ constitute the i-th *joint* of the rigid body, marked by orange rectangles and grey circles in Figure 2. The number of rigid bodies "encoded" by a single multi-body is bounded by $\Pi_{i=1}^{n}|S_i|$. A rigid body is defined by the

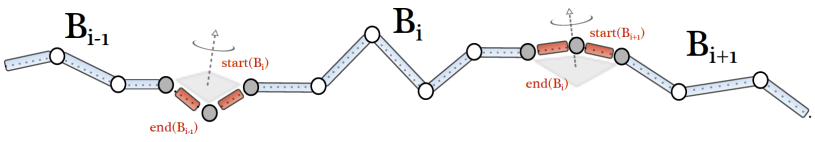

Fig. 2: A schematic representation of a rigid body

overlap of joints, and thus relies on a chain of relative roto-translations of its

blocks. Each $\mathsf{points}(B_i)$ is therefore positioned according to the coordinate system associated to a rigid block B_{i-1}. Note that once the reference system for B_1 is defined the whole rigid body is completely positioned. The relative positions of two consecutive rigid blocks B_{i-1} and B_i of a rigid body ($2 \leq i \leq n$) can be defined by a transformation matrix $T_i \in \mathbb{R}^{4\times4}$ determined from the start and end of the blocks according to the standard Denavit-Hartenberg parameters [4] obtained from the start and end of the respective blocks. We denote the product $T_1 \cdot T_2 \cdot \ldots \cdot T_i \cdot (x, y, z, 1)^T$ by $\nabla_i(x, y, z)$.

For $i = 1, \ldots, n$, the coordinate system conversion (x', y', z'), for a point $(x, y, z) \in \mathsf{points}(B_i)$ into the coordinate system of B_1, is obtained by:

$$(x', y', z', 1)^T = T_1 \cdot T_2 \cdot \ldots \cdot T_i \cdot (x, y, z, 1)^T = \nabla_i(x, y, z) \tag{1}$$

Homogeneous transformations are such that the last value of a tuple is always 1. Note that the matrix T_1 affects the positioning and rotation of the first fragment and thus the overall placement of the protein.

Definition 2 (JM-constraint). *The* joined-multibody (JM) constraint *is described by a tuple:* $J = \langle \boldsymbol{S}, \boldsymbol{V}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{E}}, \delta \rangle$*, where:*

- $\boldsymbol{S} = S_1, \ldots, S_n$ *is a multi-body. Let* $\mathcal{B} = \{B_1, \ldots, B_k\}$ *be the set of all rigid blocks in* $\boldsymbol{S}$*, i.e.,* $\mathcal{B} = \bigcup_{i=1}^{n} S_i$*.*
- $\boldsymbol{V} = V_1, \ldots, V_n$ *is a list of finite-domain variables. For* $i = 1, \ldots, n$*, the variable* V_i *is associated to a domain* $\mathsf{dom}(V_i) = \{j : B_j \in S_i\}$*.*
- $\boldsymbol{\mathcal{A}} = \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$*, and* $\boldsymbol{\mathcal{E}} = \mathcal{E}_1, \ldots, \mathcal{E}_{3n}$ *are lists of sets of 3D points such that:*
  - $\mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{A}_3$ *is the set of admissible points for* $\mathsf{start}(B)$*, with* $B \in S_1$*;*
  - $\mathcal{E}_{3i-2} \times \mathcal{E}_{3i-1} \times \mathcal{E}_{3i}$ *is the set of admissible points for* $\mathsf{end}(B)$*, with* $B \in S_i$*,* $i = 1, \ldots, n$
- δ *is a constant, used to express a minimal distance constraint between different points. Let us assume that for all* $B \in \mathcal{B}$ *and for all* $a, b \in \mathsf{points}(B)$*, if* $a \neq b$ *then* $\|a - b\| \geq \delta$ *(where* $\|\cdot\|$ *is the Euclidean norm).*

Intuitively, the JM constraint limits the spatial domains of the various blocks composing the multibody, in order to retain those fragments that assemble properly and that do not compenetrate.

A *solution* for the JM constraint J is an assignment $\sigma : \boldsymbol{V} \longrightarrow \{1, \ldots, |\mathcal{B}|\}$ s.t. there exists a sequence of matrices $T_1, \ldots, T_n$ with the following properties:
Domain: For all $i = 1, \ldots, n$, $\sigma(V_i) \in \mathsf{dom}(V_i)$.
Joint: For all $i = 1, \ldots, n-1$, let $(a^1, a^2, a^3) = \mathsf{end}(B_{\sigma(V_i)})$ and $(b^1, b^2, b^3) = \mathsf{start}(B_{\sigma(V_{i+1})})$, then it holds that (for $j = 1, 2, 3$):

$$\nabla_i(a^j_x, a^j_y, a^j_z) = \nabla_{i+1}(b^j_x, b^j_y, b^j_z)$$

Spatial Domain: Let $(a^1, a^2, a^3) = \mathsf{start}(B_{\sigma(V_1)})$, then $T_1 \cdot a^j \in \mathcal{A}_j \times \{1\}$.
For all $i = 1, \ldots, n$, let $(e^1, e^2, e^3) = \mathsf{end}(B_{\sigma(V_i)})$ then

$$\nabla_i(e^j_x, e^j_y, e^j_z) \in \mathcal{E}_{3(i-1)+j} \times \{1\}$$

where $1 \leq j \leq 3$ and $T_2, \ldots, T_i$ (in ∇_i) are the matrices that overlap $B_{\sigma(V_{i-1})}$ and $B_{\sigma(V_i)}$ (the product $\times\{1\}$ is due since we use homogeneous coordinates).

Minimal Distance: For all $j, \ell = 1, \ldots, n$, $j < \ell$, and for all points $a \in \mathsf{points}(B_{\sigma(V_j)})$ and $b \in \mathsf{points}(B_{\sigma(V_\ell)})$, it holds that:

$$\|\nabla_j(a_x, a_y, a_z) - \nabla_\ell(b_x, b_y, b_z)\| \geq \delta$$

2.1 Loop Modeling by the joined-multibody constraint

We addressed the problem of connecting two rigid block structures through a protein loop via the joined-multibody constraint. The proposed encoding and the constraint solving procedure are implemented within *FIASCO* (Fragment-based Interactive Assembly for protein Structure prediction with COnstraints) [2]. FIASCO is a C++ tool that provides a flexible environment that allows us to easily manipulate constraints targeted at protein modeling through *fragment assembly*. The starting point is a given protein together with the pair of the two known (large) blocks connected by the target loop. The model will account for them in the definitions of sets $\boldsymbol{\mathcal{E}}$. The coordinates of the initial and the final anchors, relative to the given blocks, are known. Moreover, the sequence of amino acids $a_1, \ldots, a_n$ connecting the two anchors is known. Loop modeling can be realized using the joined-multibody constraint $J = \langle \boldsymbol{S}, \boldsymbol{V}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{E}}, \delta \rangle$ where:

- For $i = 1, \ldots, n$ the set S_i contains all the protein's fragments (i.e., rigid blocks) associated with the amino acid a_i.
- $\mathsf{dom}(V_i)$ is the set of labels that uniquely identify the fragments that can be used for the amino-acid a_i. Note that a fragment is used to encode a single residue.
- The constant δ (now $\delta = 1.5$Å) asserts a minimum distance between atoms, used for overlapping fragment during each fragments assembly step.
- For the spatial domains, we set in $\mathcal{A}$ the coordinates of the initial anchor and in $\boldsymbol{\mathcal{E}}$ a 3D interval. This interval is calculated from the coordinates of the final anchor, using the covalent radii bond distances of the specific types of atoms belonging to the final anchor itself. Note that now we use intervals to represent sets of points. We use this slack for the last 3 points of the loop in order to cushion the error produced during the clustering step, still obtaining solutions that are geometrically eligible.

Let us observe that more than one loop in the same target protein can be modeled simultaneously in this way.

3 Filtering algorithm for the joined-multibody constraint

We designed a filtering algorithm (JMf) for ad-hoc propagation of the JM constraint in protein loop modeling. The Joined-Multibody filtering is inspired by arc consistency on the 3D positions of end-effectors, and uses a clustering relation over these bounds, in order to retain those domain variable assignments that produce similar spatial results. The equivalence relation captures those rigid bodies that are geometrically similar and thus compacts small differences among them; relevant gains in computation time can be derived when some errors are tolerated.

The JMf algorithm receives as input a JM-constraint $\langle \boldsymbol{S}, \boldsymbol{V}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{E}}, \delta \rangle$, a clustering function $\sim$ on the space of triples of 3D points, and a function f_{sel} that selects a representative fragment for each cluster (i.e., each equivalence class) produced by $\sim$. Note that the JMf algorithm is parametric w.r.t. $\sim$ and f_{sel}. JMf is based on an iterative procedure that computes for each body in $\boldsymbol{S}$ the fragments to be retained. Since the joints depend on the preceding bodies, the algorithm computes the domains starting from the first body. At iteration i, every fragment from S_i is joined to the previous bodies instances already computed at step i, producing a set $\mathcal{R}_i$ of end-effectors. Based on local geometric properties, the function $\sim$ computes a set of equivalence classes from $\mathcal{R}_i$. From each of such clusters, the function f_{sel} selects the new fragment representative that satisfy the constraint and that will be used to calculate $\mathcal{R}_{i+1}$ in the next step.

Clustering. The proposed clustering relation for loop modeling takes into account two factors: *(a)* the positions of the end-effectors in the 3D space and *(b)* the orientation of the planes formed by the fragments' end-effectors. This combination of clusterings allows to capture both spatial and rotational features of rigid bodies.

The spatial clustering *(a)* is based on three parameters: $k_{min}, k_{max} \in \mathbb{N}$, $(k_{min} \leq k_{max})$, and $r \in \mathbb{R}, r \geq 0$. The clustering works as follows. Given set of fragments, the end-effectors of each fragment are considered (i.e., its three last atoms) and the centroid of the triangle based on their coordinates is computed. Then, a set of k_{min} fragments, pairwise distant at least $2r$, is selected from the initial set. These fragments are selected as representatives of the equivalence classes. Other fragments will be assigned to a class whom representative centroid has mutual distance of at most r Å. This clustering ensures a rather even initial distribution of clusters, however some fragments may not fall within the k_{min} clusters. We allow to create up to $k_{max} - k_{min}$ new clusters, each of them covering a sphere of radius r. Remaining fragments are then assigned to the closest cluster.

The orientation clustering (b) partitions the fragments according to their relative orientation of the plane (called β) described by the end-effectors positions. This is handled in a pre-processing phase, being independent on other domains. The final cluster is the intersection of the two partitioning algorithms. Moreover, the representative selection function f_{sel} selects the fragments for each partition according to some preferences (e.g., most frequent fragment, closest to the center, etc.).

The filtering algorithm is similar to a directional arc consistency, when the global constraint is viewed as a conjunction of binary constraints between adjacent blocks. In particular, as soon as the domain for the variables related to the initial anchor of a JM constraint is instantiated, the corresponding constraint is woken up. The algorithm JMf is invoked with the parameters described above. If there are no empty domains after this stage, the search proceeds by selecting the leftmost variable and assigning it a fragment (block) in a leftmost order. All domains are pre-sorted from the most likely to the least likely for each variable (the previous stage of filtering preserves the ordering).

4 Experimental Results

The proposed method has been tested on a data set of 10 loop targets for each of the lengths 4, 8, and 12 residues. The targets are chosen from a set of non-redundant X-ray crystallography structures structures [3]. We analyze the performances of the Joined-Multibody filtering by examining the fraction of the search space explored during solution search, along with the qualities of the loop prediction. The latter is expressed by a meausre of the root mean square deviation (RMSD) of the atoms of proposed loop with respect to the native conformation. All experiments are conducted on a Linux Intel Core i7 860, 2.5 GHz, memory 8GB, machine.

To show the filtering power of the JM constraint we employ different evaluations based on protein loop lengths. For short protein loops (10 loops of length 4) we analyze the loop closures generated by two CSPs: the first with the JM constraint enabled *(JMf)*, and the second is a simple combinatorial fragment assembly search *(NC)*. For both problems we exhaustively explored the search space. For longer protein loops (10 loops of lengths 8 and 12, respectively), where a complete search space exploration cannot be computed in reasonable time, we compute an approximation of a filtering measure based on the ratio between the nodes pruned by propagating the JM constraint within a timeout of 600 seconds[4] and number of possible nodes expandable by an NC search.

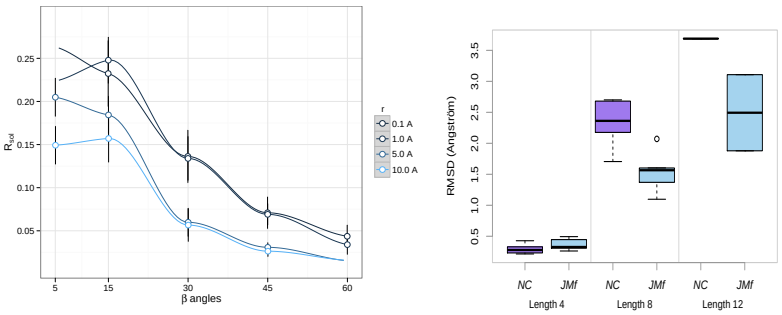

Fig. 3: Ratio of the solutions for short loop lengths (left) and RMSD comparison (right)

The aforementioned experiment are reported in Figure 3 (left) and Figure 4. We relate the filtering measures at varying of the cluster parameters r and β. The adopted parameters for the β angles are reported on the x-axes, while the r values for the clustering distances are plotted in different colors ($10\mathring{A}$ is the lightest color). The number of fragments in each variable domain is 60. This increases the likelihood to generate a loop structure that is similar to the native one.

[4] We merely count the actual search time, excluding the time spent in the clustering phase from the total running time.

Figure 3 (left) reports the ratio (R_{sol}) between the number of solutions for the CSPs with and without the JM constraint at varying of the clustering parameters, while k_{min} and k_{max} are set to 20 and 100, respectively. The white dots represent the average values of all the trials and the vertical bars illustrate the standard error of the mean: $\frac{\sigma}{\sqrt{N}}$, where σ is the standard deviation and N is the number of samples. It can be observed that the number of the solutions generated by the JM constraint decreases as the β and r values increase, as one can expect. The size of the prop-labeling tree and running times decrease with a similar trend.

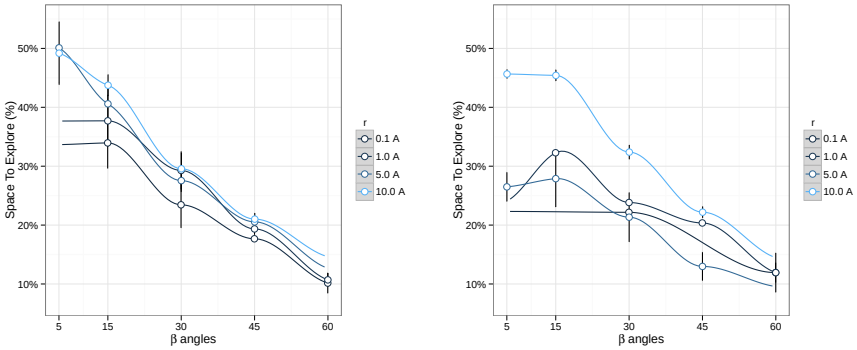

Fig. 4: Ratio of the search space explored using JMf for 8-loops (left) and 12-loops.

Longer loops are analyzed in Figure 4, which reports the percentage of the approximate space left to explore after the JM propagation. The space filtered by the JM constraint rises up to 10% of the original prop-labeling tree according to the increasing values of r and β. However, this is just an under-estimation of the pruning capability, as several fragments might immediately lead to a failure due to spatial constraints and allotting more time for the computations allows further pruning, thus increasing the filtering measure.

Figure 3 (right) reports on the qualities of the loop predictions generated by the JMf. In particular, we show that the RMSD measure is not significantly degraded by the large filtering performed by the propagator. The experiments are carried on with $k_{min} = 20$ and $k_{max} = 100$, while r and β are set respectively to 1.0 and 15 for loops of length 4, and 2.5 and 60 for longer loops. In our analysis, such parameters guarantee a good compromise between filtering power and accuracy of the results. In Figure 3 (right), the bottom and top point of each vertical line show the RMSD of the best and worst prediction, respectively, within the group of targets analyzed. The results are biased by the fragment database in use: we excluded from it the fragments that belong to the deposited protein targets. Therefore it is not possible to reconstruct the original target loop and none of the searched are expected to reach a RMSD equal to 0. The bottom and top horizontal lines on each box shows the RMSD of the 25^{th} and 75^{th} percentile prediction, respectively, while the line through the middle shows the median. We observe no substantial difference in the distributions related to

short loop predictions (length 4), and an improvement for targets of greater size due to time-out. Such results experimentally show the strength of our method: JM filtering algorithm removes successfully redundant conformations; moreover, it quickly direct the search space exploration through predictions that are biologically meaningful.

We also compare our method to three other state-of-the-art loop samplers: the Cyclic Coordinate Descent (CCD) algorithm [3], the self-organizing algorithm (SOS) [7], and the FALCm method [6]. Table 1 shows the average RMSD for the benchmarks of length 4, 8 and 12 as computed by the four programs. Note that our solution does not include specific heuristics and additional information that are used in the other programs. Moreover, it can be noted that our results are in line with those produced by the other systems, even if a general fragment database has been used in our system.

Loop	**Average RMSD**			
Length	CCD	SOS	FALCm	JMf
4	0.56	0.20	0.22	0.30
8	1.59	1.19	0.72	1.31
12	3.05	2.25	1.81	1.97

Table 1: Comparison of loop sampling methods

5 Conclusions

In this paper, we presented a novel constraint (joined-multibody) to model rigid bodies connected by joints, with constrained degrees of freedom in the 3D space, along with a filtering technique to exploit the geometrical features of the rigid bodies. We showed its application in sampling protein loop conformations. In particular, we showed that the search space of the protein loop conformations generated is drastically reduced, when the joined-multibody constraint is propagated, with controlled loss of quality.

As future work, we plan to apply our filtering method to other related applications, for example, those where there is the need of generating large number of protein conformations, by acting only on a restricted part of the protein.

Acknowledgments. The research has been partially supported by a grant from the AHPCR Center, NSF grants CBET-0754525 and IIS-0812267, and PRIN 2008 (20089M932N).

References

1. M. Ben-David, O. Noivirt-Brik, A. Paz, J. Prilusky, J. L. Sussman, and Y. Levy. Assessment of CASP8 structure predictions for template free targets. *Proteins*, 77:50–65, 2009.

2. M. Best, K. Bhattarai, F. Campeotto, A. D. Palú, H. Dang, A. Dovier, F. Fioretto, F. Fogolari, T. Le, and E. Pontelli. Introducing FIASCO: Fragment-based Interactive Assembly for protein Structure prediction with COnstraints. In *Proc. of Workshop on Constraint Based Methods for Bioinformatics*. `http://www.dmi.unipg.it/WCB11/wcb11proc.pdf`, 2011.
3. A. Canutescu and R. Dunbrack. Cyclic coordinate descent: a robotics algorithm for protein loop closure. *Protein Sci*, 12:963–972, 2003.
4. R. Hartenberg and J. Denavit. A kinematic notation for lower pair mechanisms based on matrices. *Journal of applied Mechanics*, 77:215–221, 1995.
5. L. Kinch, S. Yong Shi, Q. Cong, H. Cheng, Y. Liao, and N. V. Grishin. CASP9 assessment of free modeling target predictions. *Proteins*, 79:59–73, 2011.
6. J. Lee, D. Lee, H. Park, E. Coutsias, and C. Seok. Protein Loop Modeling by Using Fragment Assembly and Analytical Loop Closure. *Proteins*, 78(16):3428–3436, 2010.
7. P. Liu, F. Zhu, D. Rassokhin, and D. Agrafiotis. A self-organizing algorithm for modeling protein loops. *PLOS Comput Biol*, 5(8), 2009.

Qualitative Models of Cell Population Dynamics as Constraint Satisfaction Problems

Tom Kelsey[1] and Steve Linton[1]

School of Computer Science,
University of St Andrews, KY16 9SX, UK

Abstract. Existing approaches to the modelling of mammalian ovarian cell population dynamics involve differential equations. This methodology has several defects, including the inclusion of modelling assumptions that may not be supported by empirical data. We present the derivation of Constraint Satisfaction Problems based on qualitative properties of the underlying cell population dynamics. When combined with boundary conditions, exploration of the space of solutions provides a range of models that can be assessed for validity using residual errors from empirical observations. Valid models, i.e. those providing the basis for a quantitative model capable of interpreting and predicting average case population dynamics for several inter-connected cell-types, can also be categorised in terms of computational complexity of potential underlying quantitative models.

Keywords: Constraint Satisfaction, Qualitative Models, Compartmental Models, Cell Dynamics

1 Introduction

Successful computer modelling in biomedicine is a difficult undertaking. Domains are often poorly measured due to ethical, technical and/or financial constraints. In extreme instances the collection of accurate longitudinal data is simply impossible using current techniques. This adversely affects the production and assessment of hypothetical quantitative models, since the incompleteness of the domain datas necessitates the making of assumptions that may or may not reflect ground truths. A second category of assumptions are involved in the choice of quantitative modelling framework. Hypothetical solutions can be ruled out by restricting the complexity of models, and unrealistic models can be allowed by over-complex models. For both types of *a priori* assumption, mutually exclusive assumptions must be kept separate, sometimes with no biomedical justification.

One way to overcome some of these difficulties is to use qualitative modelling [12, Chapter 3], whereby the modelling process commences by describing what must, might and cannot happen in informal and conceptual terms, followed by the derivation and assessment of quantitative models that can be deployed for investigative use. Qualitative models identify the type and number of important entities of a system, and describe relationships between them in terms of

required, allowed or forbidden behaviour. Natural frameworks for such models include Constraint Programming [18], and Mixed-integer Linear (or Quadratic) Programming [9]. Each of these allows the formal description of relationships between (classes) of objects, and each has advanced techniques for either finding solutions or proving that no solution exists.

In this paper we focus on a single area of biomedicine – the maturation of human ovarian cells – and a single modelling framework – Constraint Programming. In Section 2 we describe briefly our domain of interest and limitations of the purely quantitative approach that has been used in previous studies. We describe our framework for defining classes of qualitative models and solving for candidate instances in Section 3, followed by a brief discussion in Section 4.

2 Background & Motivation

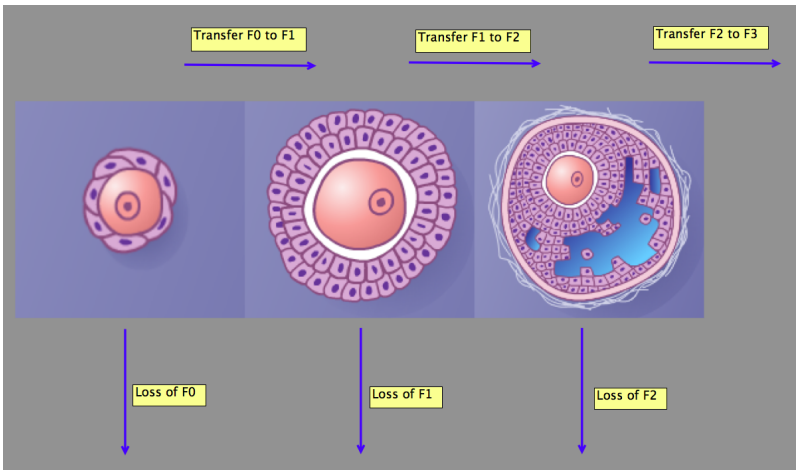

Fig. 1. Compartmental schematic of human ovarian follicular development.

The human ovary contains a population of primordial (or non-growing) follicles (F0 in Figure 1). Some of these are recruited towards maturation and start to grow. Many of these die off through atresia, but some become primary follicles (F1 in Figure 1). Again, a proportion of these die off with the remainder growing into secondary follicles (F2 in Figure 1). This continues until a very small proportion become eggs that are released from the ovary for potential fertilisation. For the purposes of this study, we consider only the dynamics of follicle progression (primordial to primary to secondary). Since there are well-defined

physiological differences between the types, the obvious choice of quantitative model is compartmental:

$$\frac{dF_0}{dt} = -k_{T_0}F_0 - k_{L_0}F_0$$

$$\frac{dF_1}{dt} = k_{T_0}F_0 - k_{T_1}F_1 - k_{L_1}F_1$$

$$\frac{dF_2}{dt} = k_{T_1}F_1 - k_{T_2}F_2 - k_{L_2}F_2$$

Kinetic loss and transfer parameters – k_{L_i} and k_{T_i} respectively – are found in principle by estimating populations at known ages, then fitting ODE solutions that minimise residual errors [8].

There are several limitations to this approach. Empirical data is scarce for primordial follicles [19], is calculated by inference for primary follicles [13], and simply does not exist for secondary follicles. Mouse-model studies have produced reasonable parameter estimates and validation [1], but it is not known how well these results translate to humans. Recent studies have shown that human ovarian stem-cells exist, suggesting that further model parameters are needed to allow for regeneration of the primordial follicle pool. The resulting models suffer from biological implausibility in the mouse model [1], and remain to be produced for humans. A key methodological drawback is that the use of compartmental models leads to a constrained class of solutions that excludes other plausible models. For example, the dynamics could also be modelled by nonlinear reaction–diffusion equations that lead to solutions that are unlikely to be obtained from a system of coupled linear ODEs (Figure 2).

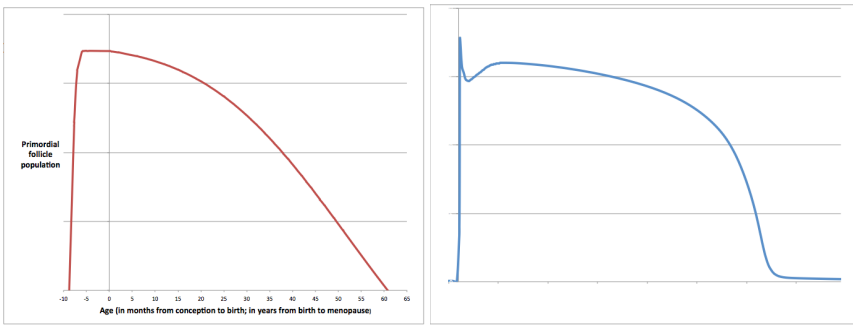

Fig. 2. Two hypothetical models of primordial follicle population from conception to menopause. On the left, a peak model adapted from [19]. On the right, the solution of a reaction–diffusion equation. Both are supported by existing physiological theory and empirical evidence, and both are intended to describe the same cell population dynamics.

3 Qualitative modelling of ovarian cell dynamics

We use the Savile Row tool that converts constraint problem models formulated in the solver-independent modelling language Essence' [10] to the input format of the Constraint Satisfaction Problem (CSP) solver Minion [11]. Savile Row converts Essence' problem instances into Minion format and applies reformulations (such as common subexpression elimination) that enhance search. As well as the standard variables and constraints expected of a CSP modelling language, Essence' allows the specification of "for all" and "exist" constraints, that are then re-cast as basic logic constraints in Minion.

We expect our candidate qualitative models to be implemented as differential equations or by non-linear curve-fitting. In both case we need to specify the notions of rate of change and smoothness. Suppose that $X[0,\ldots,n]$ is a series of variables representing a follicle population at different ages. Then we can approximate first derivatives by $X'[1,\ldots,n]$ where $X'[i] = X[i] - X[i-1]$, and second derivatives by $X''[1,\ldots,n-1]$ where $X''[j] = X'[j+1] - X'[j]$. These definitions allow us to post qualitative constraints about peak populations

$$\exists p \in [1,\ldots,n] \text{ such that } \forall i > p, X'[i] < 0 \wedge \forall i < p, X'[i] > 0.$$

We can require or forbid smoothness by restricting the absolute value of the X'' variables, and require or forbid fast rates of population growth by restrictions on the $X'[i]$.

By having three sets of variables (primordial, primary and secondary follicles) each with up to two derivative approximations, we can model interactions between the populations at different ages. For example, we can require a zero population of secondary follicles until puberty, after which the population behaviour is similar to that of primary follicles, but on a smaller scale and with an adjustable time-lag.

To further abstract away from quantitative behaviour, populations can be defined in terms of proportion of peak rather than absolute numbers of cells, different time scales can be used for different age ranges (e.g. neonatal vs postmenopausal), and we can model the qualitative behaviour of values that are normally log-adjusted in quantitative studies. Table 1 gives an illustrative example of a model involving one type of follicle.

Any solution of such a model is a candidate for the basis of a quantitative model of actual cell dynamics, once boundary conditions and scale conditions are supplied. For example, the population of each type of follicle is known to be zero at conception, and can be assumed to be below 1,000 at menopause. Several studies have indicated that peak primordial population is about 300,000 per ovary. There is initial evidence that primary follicle population peaks at 13–15 years of age in humans. Using a combination of facts and quantitative information, a range of quantitative models can be produced for later empirical validation.

Each of our qualitative models represents a class of CSPs, a set of variables with integer or Boolean domains together with a set of constraints involving

those variables. A solution is an assignment of domain values to variables such that no constraint is violated. In our methods, solutions are found by Minion using backtrack search with a variety of search heuristics. In general, there will be many more solutions to the CSP than realistic models, and many more realistic models than models that accurately describe reflect what happens in nature. Moreover, the resulting quantitative models can be graded by their complexity – linear ODE, piecewise-linear ODE, quadratic ODE, ..., non-linear PDE. Hence the ideal situation would be a CSP solution leading to an easily solved quantitive model that is biologically accurate. However, no such solution need exist, and we need to investigate the tradeoff between model complexity and model accuracy.

We can sample the space of CSP solutions by randomly ordering the variables before making value assignments, thereby constructing a different but logically equivalent search tree at each attempt. This allows us to estimate the likelihood of "good" models being found (i.e cheap and accurate), and thereby estimate the computational costs involved in attempting to find the best model that can be derived from our qualitative descriptions.

Essence' statement	Qualitative description
find $x : [int(0..max)]$ of $int(0..100)$	percentage of peak population
find y: $[int(1..max)]$ of $int(-r \cdots r)$	1st deriv. variables
find $z : int(1..max-1)]$ of $int(-r \cdots r)$	2nd deriv. variables
forall $i : int(1..max).y[i] = x[i] - x[i-1]$	1st deriv.definition
forall $j : int(1..max-1).z[j] = y[j+1] - y[j]$	2st deriv. definition
exists $k, j : int(2..birth)$.	
forall $i : int(birth..max)$.	
$i < k \Rightarrow y[i] > 0$	positive 1st deriv. pre-peak
$i > k \Rightarrow y[i] < 0$	negative 1st deriv. post-peak
$x[k] = 100 \wedge y[k] = 0$	it is a peak
$i > birth \Rightarrow \|z[i]\| < max$	smooth post-gestation

Table 1. An example of a simple qualitative model specified in Essence'. When supplied with values for max, r, and $birth$, Savile Row will construct a Minion instance, the solutions of which are all hypothetical models that respect the qualitative description.

4 Discussion

Qualitative modelling is a mature technique, with existing methods and results for qualitative compartmental models [15, 14, 16] and for the use of CSPs to describe and solve qualitative models [4, 7]. However, these latter studies either reported incomplete algorithms [4] or described complicated algebras with no associated CSP modelling language or optimised CSP solver [7]. Other approaches include process calculi and temporal logics, both of which have been shown to be successful at the molecular level [2] and the protein network level [3, 17], but not as yet at inter- and intra-cellular levels.

In this study we have utilised recent advances in CSP technology such as solver-independent modelling frameworks, specification–solver interfaces that enhance CSP instances, and the use of solvers that can quickly find all solutions to large and complex CSP instances [6, 5]. Taken together, these advances allow us to easily specify qualitative behaviour of cell dynamics, obtain solutions that generate quantitative models, and systematically investigate the tradeoffs between computational expense, model complexity and biological accuracy in a domain for which there is extremely limited direct empirical data. Our investigations utilise the search heuristics used to find CSP solutions: solvers proceed by backtrack search in a tree constructed by explicit choices for current search variable and current value assignment, by randomising these choices we can explore the space of candidate solutions.

The framework for ovarian cells treats primordial follicles as a source, and the other types as both sinks and sources. There is no feedback in the dynamical system, but we see no reason why this aspect could not be included if required. We therefore believe that this initial study can generalise to other domains at other levels of systems biology from population-based epidemiology to steered molecular dynamics.

Acknowledgments. The authors are supported by United Kingdom EPSRC grant EP/H004092/1.

References

1. Bristol-Gould, S.K., Kreeger, P.K., Selkirk, C.G., Kilen, S.M., Mayo, K.E., Shea, L.D., Woodruff, T.K.: Fate of the initial follicle pool: empirical and mathematical evidence supporting its sufficiency for adult fertility. Developmental biology 298(1), 149–54 (Oct 2006)
2. Calder, M., Hillston, J.: Process algebra modelling styles for biomolecular processes. In: Priami, C., Back, R.J., Petre, I. (eds.) Transactions on Computational Systems Biology XI, pp. 1–25. Springer-Verlag, Berlin, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-04186-0_1
3. Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S.: Machine learning biochemical networks from temporal logic properties. T. Comp. Sys. Biology pp. 68–94 (2006)
4. Clancy, D.: Qualitative simulation as a temporally-extended constraint satisfaction problem. Proc. AAAI 98 (1998)
5. Distler, A., Jefferson, C.A., Kelsey, T.W., Kotthoff, L.: The semigroups of order 10. In: Milano, M. (ed.) 18th International Conference on Principles and Practice of Constraint Programming. Springer (2012)
6. Distler, A., Kelsey, T.W.: The monoids of orders eight, nine & ten. Annals of Mathematics and Artificial Intelligence 56(1), 3–21 (Jul 2009)
7. Escrig, M.T., Cabedo, L.M., Pacheco, J., Toledo, F.: Several Models on Qualitative Motion as instances of the CSP. Revista Iberoamericana de Inteligencia Artificial 6(17), 55–71 (2002)
8. Faddy, M.J., Gosden, R.G.: A mathematical model of follicle dynamics in the human ovary. Human reproduction (Oxford, England) 10(4), 770–5 (Apr 1995)

9. Fletcher, R.: Practical Methods of Optimization, pp. 183–188. John Wiley & Sons, New York, second edn. (1987)
10. Frisch, A.M., Harvey, W., Jefferson, C., Martínez-Hernández, B., Miguel, I.: Essence: A constraint language for specifying combinatorial problems. Constraints 13(3), 268–306 (Jun 2008)
11. Gent, I.P., Jefferson, C., Miguel, I.: Minion: A fast scalable constraint solver. In: Brewka, G., Coradeschi, S., Perini, A., Traverso, P. (eds.) The European Conference on Artificial Intelligence 2006 (ECAI 06). pp. 98–102. IOS Press (2006)
12. Haefner, J.: Modeling Biological Systems. Springer-Verlag, New York (2005)
13. Kelsey, T.W., Anderson, R.A., Wright, P., Nelson, S.M., Wallace, W.H.B.: Data-driven assessment of the human ovarian reserve. Molecular human reproduction 18(2), 79–87 (Sep 2011)
14. Menzies, T., Compton, P.: Applications of abduction: hypothesis testing of neuroendocrinological qualitative compartmental models. Artificial intelligence in medicine 10(2), 145–75 (Jun 1997)
15. Menzies, T., Compton, P., Feldman, B., Toth, T.: Qualitative compartmental modelling. AAAI Technical Report SS-92-02 (1992)
16. Radke-Sharpe, N., White, K.: The role of qualitative knowledge in the formulation of compartmental models. IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews) 28(2), 272–275 (May 1998)
17. Rizk, A., Batt, G., Fages, F., Soliman, S.: Continuous valuations of temporal logic specifications with applications to parameter optimization and robustness measures. Theor. Comput. Sci. 412(26), 2827–2839 (2011)
18. Rossi, F., van Beek, P., Walsh, T.: Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier Science Inc., New York, NY, USA (2006)
19. Wallace, W.H.B., Kelsey, T.W.: Human ovarian reserve from conception to the menopause. PloS one 5(1), e8772 (Jan 2010)

Atom Mapping with Constraint Programming

Martin Mann[1], Heinz Ekker[1], Peter F. Stadler[1–5], and Christoph Flamm[1]

[1]Institute for Theoretical Chemistry, University of Vienna, Währingerstrasse 17, 1090 Vienna, Austria, [2]Bioinformatics Group, Department of Computer Science, and Interdisciplinary Center for Bioinformatics, University of Leipzig, Härtelstraße 16-18, D-04107 Leipzig, Germany, [3]Max Planck Institute for Mathematics in the Sciences, Inselstraße 22, D-04103 Leipzig, Germany, [4]Fraunhofer Institute for Cell Therapy and Immunology, Perlickstraße 1, D-04103 Leipzig, Germany, and [5]Santa Fe Institute, 1399 Hyde Park Rd., Santa Fe, NM 87501, USA
{mmann,hekker,studla,xtof}@tbi.univie.ac.at

Abstract. The mass flow in a chemical reaction network is determined by the propagation of atoms from educt to product molecules within each of the constituent chemical reactions. The Atom Mapping Problem for a given chemical reaction is the computational task of determining the correspondences of the atoms between educt and product molecules. We propose here a Constraint Programming approach to identify atom mappings for "elementary" reactions. These feature a cyclic imaginary transition state (ITS) imposing an additional strong constraint on the bijection between educt and product atoms. The ongoing work presented here identifies only chemically feasible ITSs by integrating the cyclic structure of the chemical transformation into the search.

1 Introduction

For chemical reactions often only educt and product molecules are known. The underlying mechanism, i.e., the chemical bonds that are broken or newly formed to transform the educt molecule into the product, is unknown. Equivalently, it is unknown which atom in the educt corresponds to which atom in the product. Traditionally, such knowledge is gained by isotope labeling experiments, that is, by substituting certain atoms in an educt molecule with chemically identical but physically recognizable variants that are then identified in the product molecules by means of NMR or similar methods [25]. Such approaches produce a mapping between the atoms present in the educt and product molecules and thus identify the chemical bonds that have changed. Knowledge of the reaction mechanism enables for instance the analysis and identification of metabolic pathways [3] or the classification of reactions and enzymes in terms of the mechanisms [19, 20].

The *in silico* identification of correct atom mappings is computationally non-trivial and an extensively studied task. First approaches analyzed the adjacency information within educts and products [9] using branch-and-bound search following the Principle of Minimal Chemical Distance [17] or used topological indexing based on Morgan numbering [21]. More recent methods operate directly

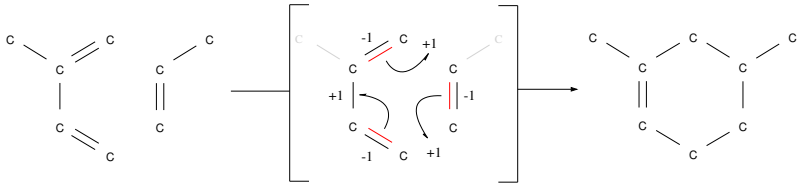

Fig. 1. Example of a Diels-Alder reaction. The ITS is an alternating cycle defined by the bonds that are broken (in red) and the bonds that are newly formed.

on graph representations of the molecules. For instance, searching for Maximum Common Edge Subgraphs (MCES) [8, 13, 14, 20, 23], an NP-hard problem, or the use of specialized energetic criteria [2, 18] allows for the identification of the static parts of the reaction and, subsequently, of the atom mapping. Another class of algorithms iteratively decomposes the molecules until only isomorphic sub-graphs remain [1, 4, 7] since it was shown by Akutsu that the MCES approaches fail for certain reactions [1].

Here, we propose a new approach to identify chemically feasible atom mappings given educt and product molecules as input. This approach makes explicit use of the observation that most reactions exhibit a cyclic transition state [16], i.e., the chemical bonds that are broken or formed are arranged in an alternating cycle. This class of mechanisms includes in particular all pericyclic reactions such as the Diels-Alder reaction, which is shown in Fig. 1 together with its transition state. We use this knowledge and focus on the identification of the cyclic *imaginary transition sub-graph* (ITS) because once identified the overall atom mapping is easily derived. For the identification of cyclic ITS candidates, constraint satisfaction problems are formulated for different cycle lengths. A fast graph matching approach is used successively to identify the overall atom mapping for each ITS solution. In the following, we will detail the problem, our constraint programming approach to identify the cyclic ITS, and how to extend an ITS candidate to a complete atom mapping for the chemical reaction.

2 Problem Definition

Given are two sets of molecules, the educts and products of a chemical reaction, each with n atoms. Both educts and products are represented by a single, not necessarily connected, undirected graph denoted $I = (V_I, E_I)$ for educts/input and $O = (V_O, E_O)$ for products/output. Each molecule corresponds to a connected component. Nodes in a molecule graph represent atoms labeled with the respective atom type $l(x)$. Following the principle of mass conservation it follows $|V_I| = |V_O|$. Edges encode covalent chemical bonds between atoms. More precisely, it is often convenient to use a multi-graph representation, in which each bonding electron pair is represented as an edge. Non-bonding electron pairs thus correspond to loops in the multi-graph. For the CSP formulation it will be more

convenient, however, to use an ordinary graph representation and to label each edge $\{x, y\} \in E_I \cup E_O$ with its bond order: single, double or triple bonds are represented by a single edge with labels 1, 2, or 3, respectively. The matrix elements $\mathcal{I}_{x,y}$ denote the number of shared bond electron pairs for the edge between the atoms x and y in the educt graph I, i.e., in practice $\mathcal{I}_{x,y} \in \{0, 1, 2, 3\}$. $\mathcal{O}$ is defined accordingly. If necessary, non-bonding electron pairs can be represented by the diagonal entries $\mathcal{I}_{x,x}$ and $\mathcal{O}_{y,y}$. Thus, the matrices $\mathcal{I}$ and $\mathcal{O}$ encode the adjacency information of the educt and product graphs, respectively.

Consider a function $\alpha : V_I \to V_O$ mapping the nodes of I onto the nodes of O and a matrix $\mathcal{Q}$ rows and columns indexed by V_I. Then we denote by $\mathcal{Q} \circ \alpha$ the matrix with entries $\mathcal{Q}_{\alpha(x),\alpha(y)}$ with rows and columns indexed by V_O. Thus $\mathcal{R}^\alpha = \mathcal{O} - (\mathcal{I} \circ \alpha)$ is well defined.

Definition. An *atom mapping* is a bijective mapping $m : V_I \to V_O$ such that

1. $\forall_{x \in V_I} : l(x) = l(m(x))$ (preservation of atom types)
2. $\mathcal{R}^m \vec{1} = 0$ (preservation of bond electrons)

The *reaction matrix* $\mathcal{R}^m$ encodes the imaginary transition state (ITS) [11, 15]. This definition of m is a slightly more formal version of the Dugundji-Ugi theory [9]. Our notation emphasizes the central role of the (not necessarily unique) bijection m. Since we consider I and O as given fixed input, the atom mapping m uniquely determines $\mathcal{R}^m$. The pair $(m, \mathcal{R}^m)$, furthermore, completely defines the chemical reaction. It therefore makes sense to associate properties of the chemical reaction directly with the atom map m.

Equivalently, the ITS can be represented as a graph $R = (V_R, E_R)$ so that E_R consists of the edges in I that are removed in O and the edges in O that were not present in I as well as the atom nodes $x \in V_R$ with at least one adjacent edge. Each edge $\{x, y\} \in E_R$ is labeled by the changes in bond order $\mathcal{R}^m_{x,y} \neq 0$. See Fig. 1 for an example. We note that in a slightly more general setting we can regard $R = (V_R, E_R)$ as a multi-graph consisting of all electron pairs that are formed or removed.

It is important to note that the existence of an atom mapping m as defined above does not necessarily imply that $\mathcal{R}^m$ is a chemically plausible ITS.

We say that two edges $\{x, y\}, \{y, z\} \in E_R$ in R are alternating if $\mathcal{R}^m_{x,y} + \mathcal{R}^m_{y,z} = 0$. A simple cycle in R of size $k > 2$ is given by the node sequence $(v_1, v_2, \ldots, v_k, v_1)$ with $v_i \in V_R$, $\{v_i, v_{i+1}\} \in E_R$, and $\forall i < j \leq k : v_i \neq v_j$. Such a simple cycle is called alternating if all successive edges as well as the ring closure $\{v_2, v_1\}, \{v_1, v_k\}$ are alternating.

Definition. An atom map m is *homovalent* if $\mathcal{R}^m_{xx} = 0$ for all $x \in V_R$. A homovalent reaction is *elementary* if its ITS R is a simple alternating cycle. Thus $\mathcal{R}^m_{x,y} \in \{-1, 0, +1\}$ holds for all elementary homovalent reactions.

In the following we outline a novel algorithm for finding atom maps for elementary homovalent reactions that is guaranteed to retrieve all possible mappings given $\mathcal{I}$, $\mathcal{O}$, and the atom labels $l(x)$ for $x \in V_I \cup V_O$.

Of course, not all I,O pairs that are educts and products of chemical transformation admit an atom mapping m with a homovalent elementary ITS. This

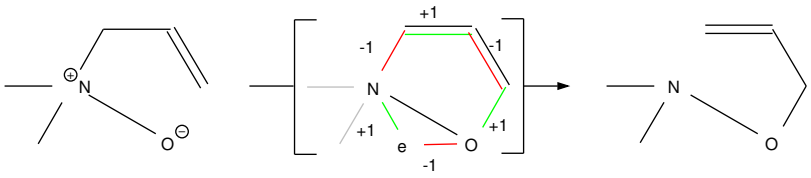

Fig. 2. The Meisenheimer rearrangement [22] transforms nitroxides to hydroxylamines. It does not admit a simple alternating cycle as ITS when molecules are represented as graphs whose vertices are atoms. An extended representation, in which the additional electron at the oxygen is treated a "pseudo-atom" can fix this issue. In such a representation an additional "charge separation" rule has to be introduced that allows an electron and a positive charge (here at the nitrogen in the product) to annihilate. This would disturbe the bijectivity of m, however.

will in general be the case for multi-step reactions and for the so-called ambivalent reactions, in which the number of non-bonding electron pairs (and thus the oxidation number of atoms) changes in the course of a reaction. Fig. 2, for example shows an example of a reaction for which it is not possible to find a simple circular ITS using the encoding above. It appears to be possible to extend the formalism outlined above also to reactions with charged atoms and radicals. This is much less well understood, however, and will require a deeper theoretical analysis in the future.

3 Constraint Programming Approach

The central problem to find an elementary homovalent atom mapping is to identify the alternating cycle defining the ITS R given the adjacency information of the educts $\mathcal{I}$ and products $\mathcal{O}$. This can be done via solving the Constraint Satisfaction Problem (CSP) as presented below. Note, due to the alternating edge condition within the ITS, we have to consider rings with an even number of atoms only. In practice, the ITS of elementary homovalent reactions involves $|V_R| = 4$, 6, or 8 atoms.

A CSP for an ITS of size $k = |V_R|$ is given by the triple (X, D, C) defining the set of variables X, according domains D_i, and the set of constraints C to be fulfilled by any solution.

We construct an explicit encoding of the atom mapping using k variables representing the ring in I and another set for the mapped nodes in O, i.e., $X = \{X_1^I, \ldots, X_k^I\} \cup \{X_1^O, \ldots, X_k^O\}$ with domains $D_i^I = V_I$ and $D_i^O = V_O$.

To find a bijective mapping we have to ensure $\forall i \neq j : X_i^I \neq X_j^I$ and $\forall i \neq j : X_i^O \neq X_j^O$, i.e., a distinct assignment of all variables. To enforce atom label preservation we need arc consistency for $l(X_i^I) = l(X_i^O)$, i.e. we have to enforce $\forall e \in D_i^I : \exists p \in D_i^O : l(e) = l(p)$ as well as $\forall p \in D_i^O : \exists e \in D_i^I : l(p) = l(e)$. Analogously, homovalence is represented by $(\mathcal{I}_{X_i^I, X_i^I} - \mathcal{O}_{X_i^O, X_i^O}) = 0$. Due to the

alternating ring condition, each atom can loose or gain at most one edge during a reaction. Thus, we can further constrain the variables with $|\,\text{degree}(X_i^I) - \text{degree}(X_i^O)| \leq 1$; where degree($v$) gives the out-degree of node v.

Finally, we have to encode the alternating cycle structure of the ITS in the mapping, i.e., for the sequence of bonds with indices 1-2-..-k-1. For all ring pair indices (i, j) we therefore require pairs with even index i to correspond the formation of a bond, i.e., we enforce $(\mathcal{O}_{X_i^O,X_j^O} - \mathcal{I}_{X_i^I,X_j^I}) = 1$, while all odd indices i are bond breaking $(\mathcal{O}_{X_i^O,X_j^O} - \mathcal{I}_{X_i^I,X_j^I}) = -1$ accordingly.

In order to avoid symmetric solutions, we introduce order constraints on the input variables: $(\forall i > 1 : X_1^I < X_i^I)$; where $X_i < X_j$ denotes $\exists (x, y) \in D_i \times D_j : x < y$ using e.g. an index order on the nodes. This ties the smallest cycle node to the first variable X_1^I and prevents the rotation-symmetric assignments of the input variables. Note, since we constrain the bond $(1, 2)$ to be a bond breaking $(\mathcal{O}_{X_1^O,X_2^O} - \mathcal{I}_{X_1^I,X_2^I} = -1)$, the direction of the cycle is fixed and all direction symmetries are excluded as well.

Although the CSP is defined above for domains of nodes $v \in V_I \cup V_O$ it can be easily reformulated using integer encodings of the atom nodes allowing the application of standard constraint solvers such as `Gecode` [12]. This enables the use of efficient propagators for most of the required constraints, such as the algorithm of Regin [24] for globally unique assignments. Only a few binary constraints, e.g. to ensure atom label preservation or the ring bonding, require a dedicated implementation, which poses no serious obstacles.

All solutions for this CSP are chemically valid ITS candidates. In order to check whether or not a true ITS is found we have to ensure that the remaining atoms, i.e., those that do not participate in the ITS, can be mapped without further bond formation or breaking. This is achieved using a standard graph matching approach as discussed in the following.

4 Overall Atom Mapping Computation

Given the CSP formulation from above, we can enumerate all valid ITS candidates for all possible ring sizes $k \in \{4, 6, 8\}$. For a CSP solution we denote with a_i^I and a_i^O the assigned values of the variables X_i^I and X_i^O, respectively. Once the ITS candidate is fixed, we can reduce the problem to a general graph isomorphism problem with a simple relabeling of the ITS edges. Thus, we derive two new adjacency matrices $\mathcal{I}'$ and $\mathcal{O}'$ from the original matrices $\mathcal{I}$ and $\mathcal{O}$, resp., as follows: For all ring pairs (i, j) within the ring sequence 1-2-..-k-1, we change the corresponding adjacency information to a unique label using $\mathcal{I}'_{a_i^I,a_j^I} = \mathcal{O}'_{a_i^O,a_j^O} \in \{f, b\}$ encoding if a bond between the mapped ITS nodes is formed (f) or broken (b). All other adjacency entries are kept the same as in $\mathcal{I}$ and $\mathcal{O}$, respectively.

Given these updated, "ITS encoding" adjacency matrices $\mathcal{I}'$ and $\mathcal{O}'$, the identification of the overall atom mapping m reduces to the graph isomorphism problem based on $\mathcal{I}'$ and $\mathcal{O}'$. Thus, all exact mappings of $\mathcal{I}'$ onto $\mathcal{O}'$ are valid atom mappings m of an elementary homovalent reaction, since the encoded ITS

respects all constraints due to the CSP formulation. The graph matching can be done using fast and efficient algorithms as the VF2-algorithm [6], which is among the fastest available [5]. Since almost all molecular graphs are planar, even faster algorithms [10] might be applicable as well.

5 Discussion

We have presented here a novel constraint programming approach to identify atom mappings for elementary homovalent reactions. The incorporation of the cyclic ITS structure within the search ensures the chemical feasibility of the mapping that is not guaranteed by standard approaches that attempt to solve Maximum Common Edge Subgraph Problems [1].

The formulation of the CSP using only the atoms involved in the ITS results in a very small CSP that can be solved efficiently. Thus, it is well placed as a filter for ITS candidates for the subsequent, computationally more expensive graph matching approaches. While not described here, the CSP could be easily extended to find the entire atom mapping by introducing additional matching variables for all atoms participating in the reaction, all constrained to preserve atom label, node degree, and bond valence information. The solutions of such an extended CSP are the desired chemically feasible atom mappings m. This involves a much larger search space, however.

At present, we consider elementary homovalent reactions only, i.e., for reactions in which the transition state is an elementary cycle with an even number of atoms. The CSP formulation can be easily extended to odd ITS cycles ($k \in \{3, 5, 7\}$), but different ring layouts have to be considered. Furthermore, such reactions are not homovalent, i.e., at least one atom participating in the ITS is gaining or losing non-bonding electrons, which requires some moderate changes in the formulation of the constraints.

Constrain programming appears to be a very promising approach to solving atom mapping problems since it provides a very flexible framework to incorporate combinatorial constraints determined by the underlying rules of chemical transformations.

References

1. T. Akutsu. Efficient extraction of mapping rules of atoms from enzymatic reaction data. *J. Comp. Biol.*, 11:449–62, 2004.
2. J. Apostolakis, O. Sacher, R. Körner, and J. Gasteiger. Automatic determination of reaction mappings and reaction center information. 2. validation on a biochemical reaction database. *J. Chem. Inf. Mod.*, 48:1190–1198, 2008.
3. M. Arita. The metabolic world of Escherichia coli is not small. *Proc. Natl. Acad. Sci. USA*, 106:1543–1547, 2004.
4. T. Blum and O Kohlbacher. Using atom mapping rules for an improved detection of relevant routes in weighted metabolic networks. *Journal of Computational Biology*, 15:565–576, 2008.

5. L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. Performance evaluation of the VF graph matching algorithm. In *Proceedings of the 10th International Conference on Image Analysis and Processing*, ICIAP '99, page 1172. IEEE Computer Society, 1999.
6. L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(10):1367–72, 2004.
7. J. D. Crabtree and D. P. Mehta. Automated reaction mapping. *J. Exp. Algor.*, 13:1.15–1.29, 2009.
8. M. J. L. de Groot, R. J. P. van Berlo, W. A. van Winden, P. J. T. Verheijen, M. J. T. Reinders, and D. de Ridder. Metabolite and reaction inference based on enzyme specificities. *Bioinformatics*, 25(22):2975–83, 2009.
9. James Dugundji and Ivar Ugi. An algebraic model of constitutional chemistry as a basis for chemical computer programs. *Topics Cur. Chem.*, 39:19–64, 1973.
10. D. Eppstein. Subgraph isomorphism in planar graphs and related problems. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, SODA '95, pages 632–40. Society for Industrial and Applied Mathematics, 1995.
11. S Fujita. Description of organic reactions based on imaginary transition structures. 1. introduction of new concepts. *J. Chem. Inf. Comput. Sci.*, 26:205–212, 1986.
12. Gecode: Generic constraint development environment, 2007. Available as an open-source library from `www.gecode.org`.
13. M. Hattori, Y. Okuno, S. Goto, and M. Kanehisa. Heuristics for chemical compound matching. *Genome Informatics*, 14:144–53, 2003.
14. M. Heinonen, S. Lappalainen, T. Mielikäinen, and J. Rousu. Computing atom mappings for biochemical reactions without subgraph isomorphism. *J. Comp. Biol.*, 18:43–58, 2011.
15. J B Hendrickson. Comprehensive system for classification and nomenclature of organic reactions. *J Chem Inf Comput Sci*, 37:852–860, 1997.
16. Rainer Herges. Organizing principle of complex reactions and theory of coarctate transition states. *Angewante Chemie Int Ed*, 33:255–276, 1994.
17. C. Jochum, J. Gasteiger, and I. Ugi. The principle of minimum chemical distance (PMCD). *Angew. Chem. Int. Ed.*, 19:495–505, 1980.
18. R. Körner and J. Apostolakis. Automatic determination of reaction mappings and reaction center information. 1. the imaginary transition state energy approach. *J. Chem. Inf. Mod.*, 48:1181–1189, 2008.
19. M. Kotera, Y. Okuno, M. Hattori, S. Goto, and M. Kanehisa. Computational assignment of the EC numbers for genomic-scale analysis of enzymatic reactions. *J. Am. Chem. Soc.*, 126:16487–16498, 2004.
20. M. Leber, V. Egelhofer, I. Schomburg, and D. Schomburg. Automatic assignment of reaction operators to enzymatic reactions. *Bioinformatics*, 25:3135–3142, 2009.
21. M. Lynch and P. Willett. The automatic detection of chemical reaction sites. *Journal of Chemical Information and Computer Sciences*, 18:154–159, 1978.
22. Jakob Meisenheimer. Über eine eigenartige Umlagerung des Methyl-allyl-anilin-N-oxyds. *Chemische Berichte*, 52:1667–1677, 1919.
23. J. W. Raymond and P. Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *J. Computer-Aided Mol. Design*, 16:521–33, 2002.
24. J.-C. Regin. A filtering algorithm for constraints of difference. In *Proceedings of the 12th National Conference of the American Association for Artificial Intelligence*, pages 362–367, 1994.
25. W. Wiechert. ^{13}C metabolic flux analysis. *Metabolic Engineering*, 3:195–206, 2001.

An ASP Implementation of PhyloWS

Tiep Le, Hieu Nguyen, Enrico Pontelli, and Tran Cao Son

Department of Computer Science, New Mexico State University
(tile,nhieu,epontell,tson)@cs.nmsu.edu

Abstract. This paper illustrates the use of ASP technology in implementing the PhyloWS web service API—a recently proposed and community-agreed standard API to enable uniform access and interoperation among phylogenetic applications and repositories. To date, only very incomplete implementations of PhyloWS have been realized; this paper demonstrates how ASP provides an ideal technology to support a more comprehensive realization of PhyloWS on a repository of semantically-described phylogenetic studies. The paper also presents a challenge for the developers of ASP-solvers.

1 Introduction

Phylogenetic inference is the task of constructing a phylogenetic tree that accurately characterizes the evolutionary lineages among a set of given species or genes. Phylogenetic trees allow us to understand the lineages of various species and how various functions evolved, to inform multiple alignments, and to identify what is the most conserved or important in some class of sequences. As such, phylogenetic trees have gained a central role in modern biology. They have become fundamental tools for building new knowledge, thanks to their explanatory and comparative-based predictive capabilities [9].

The explosive growth of phylogenetic data and the central role of phylogenetic knowledge in system biology led to the development of a database of phylogenies, called TreeBASE (e.g., [15, 19], `www.treebase.org`). The database contains phylogenetic trees and data matrices, together with information about the relevant publication, taxa, morphological and sequence-based characters, and published analyses. The trees are stored as text field strings structured in the Newick format [8]. The database provides retrieval capabilities via a web interface, allowing users to locate phylogenies and to obtain datasets for different studies. Users can also retrieve data via a web service interface API (`sourceforge.net/apps/mediawiki/treebase/index.php?title=API`). This interface can deliver data in several different formats, including Newick, NEXUS [13], JSON, NeXML [22].

The creation of TreeBASE is a significant step towards the goal of creating the Tree of Life. Yet, it has been recognized that the lack of interoperability and standards in data and services between tools for the inference of phylogenies prevent large-scale and integrative analyses. To address these shortcomings, several efforts have been made. One of such efforts led to the development of an *interoperation stack* (*EvoIO Stack*) for the encoding and exchange of evolutionary

structures. EvoIO comprises of *(i)* an ontology for data description (*Comparative Data Analysis Ontology (CDAO)*) [18], *(ii)* an exchange format (*NeXML*) [22], and *(iii)* a web service interface (*PhyloWS*) [11].

The PhyloWS interface specification [11] identifies several classes of queries specifically tied to phylogenies. The interface is comprehensive and represents the most extensive collection of queries and transformations for biological phylogenies proposed to-date—in particular, it largely subsumes previous attempts to characterize access to phylogenetic databases (e.g., the approach of [16], implemented in Prolog by [3]). The implementation of these queries on an RDF representation of phylogenies proved to be challenging; in particular, traditional languages for RDF (e.g., SPARQL) do not provide the power to perform the type of computations on phylogenies required by PhyloWS—e.g., they lack the expressive power to capture recursive computations and transitive closures (which are essential, e.g., to determine ancestors and lineage in a phylogeny).

Contribution: In this paper, we propose a modular implementation of PhyloWS using answer set programming (ASP) [14]. We present the encoding of PhyloWS, which shows that ASP is ideal to answer various types queries from the PhyloWS specification. Experimental results are presented in a companion paper [12].

The overall implementation of PhyloWS is depicted in Figure 1. The top part shows the components of the system necessary to populate CDAOStore; CDAOStore [3] is a triple store, built using CDAO and used for our experiments. First, data from current phylogenetic tree repositories (e.g., TreeBASE) is extracted into NeXML data files (*Extractor*) and converted to CDAO representation (*Converter*). This process is executed only once to populate the repository of phylogenetic data in CDAO representation. A standard XML-parser is used to generate triples from NeXML and import them into the CDAOStore.

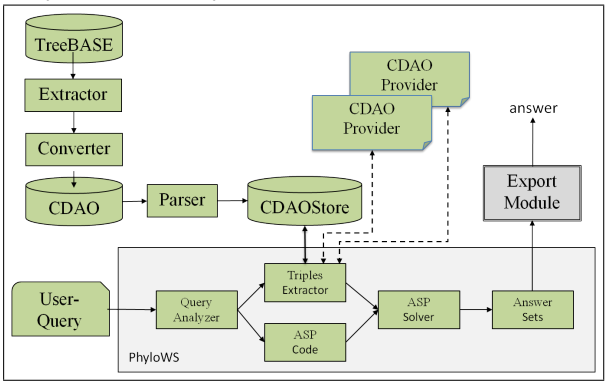

Fig. 1: Overall Structure: System Implementation

The main contribution of this paper is the PhyloWS box. User queries are analyzed by a query analyzer that determines the actual ASP-code and the necessary data type from the CDAOStore repository. This information is passed on to the *Triples Extractor* module. The ASP program (facts and code) is sent to the ASP solver to compute its answer sets. The export module obtains the answer sets from the solver and generates the answer for the user. Let us emphasize that the PhyloWS implementation in this paper is fully modular, and can be applied to any data source that can provide phylogenetic data as CDAO RDF triples. Note, that in the construction of the CDAOStore, we were able to reuse the pro-

totype described in [3]. The present system includes several improvements over the reported prototype—e.g., it is able to extract *all* studies from TreeBASE, the conversion to CDAO is more precise.

2 PhyloWS in ASP

PhyloWS [11] is a web-services standard for accessing phylogenetic trees, data matrices, and their associated metadata from online phylogenetic data. Together with NeXML and CDAO, PhyloWS is a part of the platform, called *EvoIO Stack* [20], that combines support for exchange of data and their semantics and predictable programmatic access. To date, PhyloWS only exists as a specification. The implementation proposed in this paper is its first implementation[1]. PhyloWS contains specification for a variety of tasks necessary for the creation, maintaining, retrieving, and manipulating of phylogenetic data (e.g., trees, matrices). In this paper, we will focus on the services for retrieving phylogenetic data. For efficiency purpose, we develop a front-end that analyzes the queries and determines the type of phylogenetic data that needs to be extracted from the CDAO repository. Presently, the type of data corresponds to the type of queries. For example, for a node-oriented query, information about the trees (nodes and edges) will be extracted. The *Triples Extractor* module is responsible for extracting the necessary information from the CDAO representation and generating ASP facts for use to answer the query. Observe that this task can be combined and executed via ASP extensions such as `dlvhex` [4]. We will discuss the reason behind our design choice in the discussion part of the paper.

Representing CDAO in ASP. The information about phylogenies can be easily encoded as ASP facts. Following are some sample facts generated by the *Triples Extractor* module:

```
tree(t_id).                      % t_id is a tree
tree_is_defined_by(t_id,s_id).   % t_id is studied in s_id
tree_ntax(t_id,n_Taxa).          % t_id has n_Taxa taxa
edge(t_id,n1,n2).                % t_id contains an edge  from n1 to n2
edge_length(t_id,n1,n2,l).       % l =  length of the edge (n1,n2) in t_id
represents_TU(t_id,n1,tu_id).    % node n1 of t_id represents tu_id
taxon_id(tu_id,taxon_id).        % tu_id represents taxon_id
matrix_type(m_id,m_type).        % matrix m_id is of the type "m_type"
belongs_to_TU(m_id,cell,tu_id). % cell in matrix m_id belongs to tu_id
```

ASP Encoding of PhyloWS. The encoding of the PhyloWS in ASP starts with the definition of a set of rules that will be frequently used in several types of queries. The rules defined common concepts like `node(T,N)` (`N` is a node of tree `T`), `parent(T,N1,N2)` (`N1` is the parent of `N2` in tree `T`), etc. It also defines the predicate `common_ancestor` of a set of taxa, identified by the predicate `set_of_taxa`, whose elements will be specified by `member/2`.

[1] The implementation of PhyloWS at `sourceforge.net/apps/mediawiki/treebase/index.php?title=API` is tailored to TreeBASE and limited to simple retrievals.

Node-oriented Queries. We currently consider four frequently used node-oriented queries.

- *Query N1*: *compute the most recent common ancestor of two or more leaf nodes in a specified tree.* The input consists of a tree t and a set s of leaf nodes in t. The output should be the most recent common ancestor n of elements in s, denoted by $mrca(n, s)$, determined using the ASP rule:

```
mrca(N, S):- tree(T), node(T,N), set_of_taxa(S),
             common_ancestor(T,N,S),
             {common_ancestor(T,Nb,S): ancestor(T,N,Nb)}0.
```

- *Query N2*: *compute the patristic distance between two taxa in a given tree.* The patristic distance between taxa n_1 and n_2 of a tree t is defined as the sum of the distances from the most recent common ancestor to each node:

```
set_of_taxa(s). member(n1,s). member(n2, s).
distance_to_ancestor(T,N1,N2,L):- parent(T,N1,N2),
                                  edge_length(T,N1,N2,L).
distance_to_ancestor(T,N1,N2,D):- parent(T,Nb,N2),
                                  edge_length(T,Nb,N2,L),
        distance_to_ancestor(T,N1,Nb,L2), D=L+L2.
patristic_distance(T,N1,N2,D):- mrca(M, s), D=L1+L2,
        distance_to_ancestor(T,M,n1,L1),
        distance_to_ancestor(T,M,n2,L2).
```

  The rules are simple thanks to the definition of the most recent common ancestor of a set in *Query N1*. Rules for computing the distance are standard.

- *Query N3*: *identify the set of* matching nodes *of a tree whose distance to the root is greater than a predefined distance.* Given a tree t (e.g., by identifier) and a distance c, output the matching nodes whose distance to the root is greater than c. This is implemented by the following rule:

```
matching_nodes(T,N):- root(T,R), distance_to_ancestor(T,R,N,L), L>=c.
```

- *Query N4*: *compute the lineage of ancestors of a node.* Given a tree t, a node n, the lineage of ancestors for n can be determine by the following rule, built using the facts `has_Ancestor(t, n, x)`, i.e., x is an ancestor of n in t.

```
lineage_node(T,N,Ancestor_id) :- has_Ancestor(T,N,Ancestor_id).
```

Clade-oriented Queries. Two typical clade-oriented queries are implemented.

- *Query C1*: *find the minimum spanning clade and the taxonomic units (TUs) of the clade for a set of taxa in a specified tree.* Given a set of taxa s, determine the minimum spanning clade of s and its TUs. Nodes belong to the minimum spanning clade are represented by the atoms of the form $minimum\_clade(s, n)$. Atoms of the form $label(x, y)$ represent the label associated to the nodes in the clade. Since the answer is the tree whose root is the `mrca` n of s and all n's descendants, this can be implemented as follows.

```
minimum_clade(S,N):- tree(T), node(T,N), mrca(N, S).
minimum_clade(S,D):- tree(T), node(T,N), mrca(N, S), ancestor(T,N,D).
label(D,TU_Label):- tree(T), minimum_clade(_,D),
                    represents_TU(T,D,TU),tu_label(T,TU,TU_Label).
```

The first rule states that the given most recent common ancestor belongs to the minimum clade. The next rule obtains all of its descendants.

- *Query C2*: *find a clade in a tree whose taxa has a given character*, i.e., given a tree t and a character c, find a clade (or all) s of t s.t. every taxon in s has the character c.

```
clade(s).                          {in_clade(s,N) : leaf(t,N)}.
member(N,s):- in_clade(s,N).
:- minimum_clade(s, N),not in_clade(s, N).
:- in_clade(s,N), represents_TU(T,N,TU),
        belongs_to_TU(M,Cell,TU), not belongs_to_Character(M,Cell,c).
```

The fact $clade(s)$ specifies the name s of the clade produced. The choice rule states that a leaf might or might not belong to the clade. The third rule defines the membership of the node in the set of taxa s that has been selected to determine the minimum clade (Query C1). The first constraint ensures that the elements of the clade are only those that are selected. The second removes clades that contain taxa that do not have the specified character.

Tree-oriented Queries. We consider eight types of tree-oriented queries.

- *Query T1*: *find trees matching a topology.* The topology can be given by (*i*) the range of the numbers of taxa count of the tree, i.e., between $n - c$ and $n + c$ for two constants n and c; (*ii*) the range of the width of the tree; etc. Most of the above queries can be straightforwardly encoded in ASP. For example, given a constant c and the tree $tr13$, the following rule determines all trees with their taxa count in the range $[n - c, n + c]$ where n is the number of taxa of $tr13$. The rule makes use of the predicate $tree\_ntax(t, n)$ that represents the number of taxa of a tree.

```
matching_ntax(T,Cnt):- tree_ntax(tr13,N),tree_ntax(T,Cnt),
                       Cnt <= N+c,Cnt>=N-c.
```

- *Query T2*: *find trees whose length is shorter (or longer) than the length of a given tree (or a constant)* where the tree length is defined as the maximal distance from the root of the tree to its taxa (leaves). This type of queries can be answered with the definition of the tree length, implemented as follows.

```
distance_to_root(T,N,L):- root(T,R),leaf(T,N),
                          distance_to_ancestor(T,R,N,L).
tree_length(T,L):- root(T,R), leaf(T,N), distance_to_root(T,N,L),
                          {distance_to_root(T,X,L1): L1>L}0.
```

- *Query T3*: *find trees with the shortest distance from the root to a given node* n. This can be easily implemented using the predicate *distance_to_root*.
- *Query T4*: *given a set s of OTUs (or taxa), find a tree containing this set.* We present the rules for identifying trees with a set of OTU, specified by the atom $otu\_set(s)$ and the membership atoms $member(x, s)$.

```
connect_tu(TU,S,T):- tree(T),otu_set(S),member(TU,S),
                     represents_TU(T,_,TU).
tree_otus(T,S):- tree(T),otu_set(S),
                     {member(TU,S):not connect_tu(TU,S,T)}0.
```

- *Query T5*: *find trees based on tree metadata*. For example, trees that were *(i)* created by some author; *(ii)* created before a given date; *(iii)* built with a given type of data; etc. Since the data included in each study contains information such as *has_creator*, *has_creationDate*, *matrix_type*, etc. this type of queries can be implemented in an obvious manner.
- *Query T6*: *identify trees by the parsimony tree length* which is defined by the total number of characters of its taxa. This can be implemented as follows.

```
parsimony_length(T,L):- tree(T),
        L =  #count {belongs_to_Character(_,Cell_id,Character):
        belongs_to_TU(_,Cell_id,TU_id):represents_TU(T,_,TU_id)}.
```

- *Query T7*: *determine trees with size greater (or smaller) than a given constant c, or a certain ratio r of internal to external nodes*. Using the aggregate function *#count*, this type of query can be implemented as follows[2].

```
matching_tree_size(T,S):- tree(T), S = #count {node(T,_)}, S>=c.
internal_node(T,N):- node(T,N), not leaf(T,N).
matching_tree_ratio(T):- tree(T),R=R1/R2, R>=r,
        R1 = #count {internal_node(T,_)}, R2 = #count {leaf(T,_)}.
```

- *Query T8*: *computing the Robinson-Foulds distance [1] between two trees*. The Robinson-Foulds distance is frequently used to compare phylogenetic trees. It measures the number of clusters of descendant leaves that are not shared by the two trees. The Robinson-Foulds distance of two trees T_1 and T_2 can be computed using the following algorithm:
  - Compute the multi-set of clusters of each tree, where each cluster is a set of taxa (leaves) that are descendants of an internal node n. Let us denote the set of clusters of T_1 and T_2 by $C(T_1)$ and $C(T_2)$, respectively.
  - Compute D_1 (resp. D_2), the number of clusters which belong $C(T_1) \setminus C(T_2)$ (resp. $C(T_2) \setminus C(T_1)$).

  The Robinson-Foulds distance is then defined by $(D_1 + D_2)/2$. Given two trees T_1 and T_2, we define the predicate $rf\_distance(T_1, T_2, D_1, D_2)$ that encodes the Robinson-Foulds distance. This can be implemented using the following set of ASP rules.

```
in_cluster(T,N,L):- internal(T,N), leaf(T,L), ancestor(T,N,L).
neq_cluster(X,Y):- internal(T1,X),internal(T2,Y),T1!=T2,i
            n_cluster(T1,X,L), not in_cluster(T2,Y,L).
neq_cluster(X,Y):- internal(T1,X),internal(T2,Y),T1!=T2,
            not in_cluster(T1,X,L),in_cluster(T2,Y,L).
eq_cluster(X,Y,T1,T2):- internal(T1,X),internal(T2,Y),T1!=T2,
            not neq_cluster(X,Y).
{matched(X,Y,T1,T2) : eq_cluster(X,Y,T1,T2)}.
2{used(T1,X), used(T2,Y)}:- matched(X,Y,T1,T2).
matched(X,Y,T1,T2):- matched(Y,X,T2,T1).
:-matched(X,Y,T1,T2),matched(X,Z,T1,T2),Y!=Z.
:-matched(Y,X,T1,T2),matched(Z,X,T1,T2),Y!=Z.
```

[2] The code assumes that the ratio is an integer. Using the scripting feature available for `clingo`, this assumption can be removed.

```
:-eq_cluster(X,Y,T1,T2), not used(T1,X), not used(T2,Y).
not_matched(T,N):- internal(T,N),not used(T,N).
rf_distance(T1,T2,D1,D2):- tree(T1), tree(T2), T1!=T2,
     D1 = #count {not_matched(T1,N)}, D2 = #count{not_matched(T2,N)}.
```

The clusters are named by the internal nodes. The first rule defines the elements of a cluster. Next two rules state that two clusters from different trees are different when their sets of taxa are different. The third rule defines when two clusters are identical. The choice rule defines the predicate $matched(X, Y, T_1, T_2)$ among identical clusters of the trees. The next rules define the predicates $matched(X, Y, T_1, T_2)$, $used(X, T_1)$, and $used(Y, T_2)$, indicating that the cluster X of tree T_1 is identical to the cluster Y of tree T_2 and will not be counted towards D_1 and D_2 respectively. The constraints ensure that each cluster is used to match with at most one cluster and the matching should be done as long as it is possible. $not\_matched(T, N)$ indicates the cluster that is not matched with any cluster of another tree. The last rule encodes the Robinson-Foulds distance.

Data-oriented Queries. We consider four types of data-oriented queries.

- *Query D1*: *list metadata for a given taxon n or a given tree t*. Such information is available from the facts associated to the tree, e.g.,

```
metadata_belongs_to(N,T,Study_id):- node(T,N),
        tree_is_defined_by(T,Study_id).
metadata_represents(N,TUId,TULabel,TaxonId,TaxonVariant,Ncbi,Ubio):-
        represents_TU(T,N,TUId), tu_label(T,TUId,TULabel),
        taxon_id(TUId,TaxonId),taxonVariant_id(TUId,TaxonVariant),
        ncbi_id(TUId,Ncbi), ubio_id(TUId,Ubio).
metadata_character(N,Character_id):-
        represents_TU(_,N,TU_id),belongs_to_TU(_,Cell_id,Tu_id),
        belongs_to_Character(_,Cell_id,Character_id).
```

- *Query D2*: *identify all matrices containing a given OTU (tu_id); or determine all characters in a matrix (m_id) that have data for an OTU*. This query is encoded as follows.

```
matrices_with_otu(M,tu_id):- has_TU(M,tu_id).
character_has_otu(C,m_id,tu_id):- belongs_to_TU(m_id,Cell,tu_id),
                                  belongs_to_Character(m_id,Cell,C).
```

- *Query D3*: *identify all OTUs in a matrix which have a given set of characters*:

```
has_character(Tu,C):- belongs_to_TU(M,Cell,Tu),
        belongs_to_Character(M,Cell,C).
obtain_otus_having_characters(Tu,S):- has_TU(M,Tu),set_characters(S),
        {member(C,S): not has_character(Tu,C)}0.
obtain_otus_having_characters_belonging_matrix(matrix_id,Tu,S):-
        has_TU(M,Tu), set_characters(S),
        {member(C,S): not has_character(Tu,C)}0.
```

- *Query D4*: *identify the character that appears in all matrices containing data for a given set of OTUs*. The ASP rules for this query are:

```
matching_matrices(M,S):- otu_set(S), has_TU(M,_),
```

```
        {member(E,S): not has_TU(M,E)}0.
has_character(M,C):- belongs_to_Character(M,_,C).
character_in_all_matrices(C):- matching_matrices(M,S),
        has_character(M,C),
        {matching_matrices(M1,S): not has_character(M1,C)}0.
```

The first rule identifies the matrix that contains all the given OTUs. The other rules search for the characters that appear in all those matrices.

3 Discussion and Conclusions

Design Choices. ASP technologies have been extended to allow ASP programs interact with ontologies such as the system `dlvhex` [4]. As such, it is natural to ask the question of whether PhyloWS could be implemented using `dlvhex` and how would the system perform. To answer these questions, we have experimented with the web interface at `http://asptut.gibbi.com/`. With a few changes in the syntax to conform with the `dlvhex` syntax, most queries can be executed with sample data. The difficulty arises when we attempt to run with the real data. As it turns out, converting everything into triples using `dlvhex` using the command: `triple(X,Y,Z):-&rdf[file_URI](X,Y,Z)` and then defining necessary predicates such as `has_TU, belongs_to_TU` using LP rules such as[3]

```
has_TU(X,Z) :- triple(X,"<http://___/cdao.owl#has_TU>",Z).
belongs_to_TU(X1,Y1,Z1) :-
    triple(X1,"<http://___/cdao.owl#has_Character>",Z2),
    triple(Y1,"<http://___/cdao.owl#belongs_to_Character>",Z2),
    triple(Y1,"<http://___/cdao.owl#belongs_to_TU>", Z1).
```

does not provide the desired efficiency. For example, our parser took 30 minutes to process the study `S261` (12 MB in CDAO representation); the web-interface does not return the result after 1.5 hours. This indicates that a straightforward application of `dlvhex` features to simplify the amount of programming will not yield an acceptable result. We are planning to further experiment with `dlvhex` without using the web-interface.

The huge size of the CDAO files and the lack of an efficient interface between ASP and ontologies led to the use of the parser (using JAVA and the Jena framework) to generate facts from CDAO and store them in the CDAOStore.

As noted, the current size of the CDAOStore is about 5GB. Intuitively, any query listed in Section 2 could have been processed using this data. However, `clingo` cannot deal with file larger than 70 MB. We observed this during our experiment: whenever the amount of data is more than 70MB, a *killed* message is displayed and the computation is aborted. The *Query Analyzer* and *Triple Extractor* modules are developed to deal with this issue.

Limitations and Challenges. The previous discussion details some limitations of the current system. While it would be interesting whether the use of `dlvhex`

[3] `___` stands for `www.evolutionaryontology.org/cdao/1.0`.

will help us to eliminate the intermediate steps of the *Query Analyzer* and *Triple Extractor* modules, the critical limitation lies in the scalability of ASP-solver. As we have mentioned, `clingo` cannot yet deal with input larger than 70MB. Considering that in the current experiment, we only use data from 261 studies (around 1/10 of the total number of studies) and the necessary data could go up to 44MB, a full implementation of PhyloWS using ASP will require additional techniques and better solvers. This also raises the question of whether other ASP extensions (e.g., DLVDB [21]) provide a more scalable implementation.

Conclusion and Future Work. We described an ASP based implementation of PhyloWS, a web services API for phylogenetic applications. The implementation focuses on retrieval services, expressed by four different types of queries. We discussed the ASP implementation of the queries and evaluated with data from 261 studies extracted from TreeBASE. We detailed the design choices and discussed its limitations, that presents a challenge to the ASP community.

To continue with the development of PhyloWS, we plan to exploit the strengths of ASP to enrich PhyloWS with (*i*) constraints over the answers; and (*ii*) preferences between answers. We envision that this can be achieved via a web-interface that not only allows users to specify their queries but also the additional constraints and preferences. We plan to experiment with other ASP-extensions such as `dlvhex` or DLVDB to identify a more scalable system. In addition, we will also investigate whether different methods of computing answer sets (e.g., using reactive answer set solver) could be useful. Finally, we plan to complete the import of data from the 9558 CDAO files to CDAOStore.

References

1. T. Asano, J. Jansson, K. Sadakane, R. Uehara, and G. Valiente. Faster computation of the Robinson-Foulds distance between phylogenetic networks, *Combinatorial Pattern Matching*, Springer Verlag, 2010.
2. O. R. P. Bininda-Emonds. *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life,* Computational Biology Series, Vol. 4. Kluwer Academic Publisher, 2004.
3. Brandon Chisham, Enrico Pontelli, Tran Cao Son, and Ben Wright. Cdaostore: A phylogenetic repository using logic programming and web services. In John P. Gallagher and Michael Gelfond, editors, *Technical Communications of the 27th International Conference on Logic Programming, ICLP 2011, July 6-10, 2011, Lexington, Kentucky, USA*, volume 11 of *LIPIcs*, pages 209–219. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.
4. Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. Acm international conference on web intelligence. In *Web Intelligence*, pages 1073–1074. IEEE Computer Society, 2006.
5. H. Ellegren. Comparative genomics and the study of evolution by natural selection. *Molecular Ecology*, 17(21):4586–4596, 2008.
6. Esra Erdem. PHYLO-ASP: Phylogenetic Systematics with Answer Set Programming. In Esra Erdem, Fangzhen Lin, and Torsten Schaub, editors, *Proceedings of the 10th Logic Programming and Nonmonotonic Reasoning, 10th International Conference, Potsdam, Germany, September 14-18*, volume 5753 of *Lecture Notes in Computer Science*, pages 567–572. Springer, 2009.

7. Esra Erdem, Vladimir Lifschitz, and Donald Ringe. Temporal phylogenetic networks and logic programming. *TPLP*, 6(5):539–558, 2006.
8. J. Felsenstein. The newick tree format, 1986. `http://evolution.genetics.washington.edu/phylip/newicktree.html`.
9. W. M. Fitch. Uses for evolutionary tree. *Phi. Trans. R. Soc. Lond. B*, 349:93–102, 1995.
10. M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. clasp: A conflict-driven answer set solver. In C. Baral, G. Brewka, and J. Schlipf, editors, *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*, volume 4483 of *Lecture Notes in Artificial Intelligence*, pages 260–265. Springer-Verlag, 2007.
11. H. Lapp and R. Vos. Phyloinformatics web services API: Overview. `https://www.nescent.org/wg/evoinfo/index.php?title=PhyloWS`, National Evolutionary Synthesis Center, 2009.
12. T. Le, H. Nguyen, E. Pontelli, and T. Son. ASP At Work: An ASP Implementation of PhyloWS. In *International Conference on Logic Programming*, 2012.
13. D. Maddison, D. Swofford, and W. Maddison. NEXUS: an Extensible File Format for Systematic Information. *Syst. Biol.*, 46(4):590–621, 1997.
14. V. Marek and M. Truszczyński. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-year Perspective*, pages 375–398, 1999.
15. V. Morell. TreeBASE: the roots of phylogeny. *Science*, pages 273–569, 1996.
16. L. Nakhleh, D. Miranker, F. Barbancon, W. Piel, and M. Donoghue. Requirements of phylogenetic databases. In *Third IEEE Symposium on Bioinformatics and Bioengineering*, pages 141–148. IEEE, 2003.
17. I. Niemelä. Logic programming with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3,4):241–273, 1999.
18. F. Prosdocimi, B. Chisham, E. Pontelli, J.D. Thompson, and A. Stoltzfus. Initial implementation of a comparative data analysis ontology. *Evolutionary Bioinformatics*, 5:47–66, July 2009.
19. M. Sanderson, B. G. Baldwin, G. Bharathan, C. S. Campbell, D. Ferguson, J. M. Porter, C. VonDohlen, M. F. Wojciechowski, and M. J. Donoghue. The growth of phylogenetic information and the need for a phylogenetic database. *Syst. Biol.*, 42:562–568, 1993.
20. A. Stoltzfus, N. Cellinese, K. Cranston, H. Lapp, S. McKay, E. Pontelli, and R. Vos. The evoio interop project. `http://www.evoio.org/wiki/Main_Page`, National Evolutionary Synthesis Center, 2009.
21. Giorgio Terracina, Nicola Leone, Vincenzino Lio, and Claudio Panetta. Experimenting with recursive queries in database and logic programming systems. *TPLP*, 8(2):129–165, 2008.
22. R. Vos. `nexml`: Phylogenetic data in xml. `http://www.nexml.org`, 2008.
23. C. Webb, D. Ackerly, M. McPeek, and M. Donoghue. Phylogenies and communtiy ecology. *Annu. Rev. Ecol. Syst.*, 33(1), 2002.

Author Index