Using Planning for Query Decomposition in Bioinformatics

Biplav Srivastava
Email: sbiplav@in.ibm.com
IBM India Research Laboratory
Block 1, IIT Delhi, Hauz Khas, New Delhi 110016, India.

Abstract

Domains like bioinformatics are *complex* data integration domains because data from remote sources *and specialized applications* need to be combined to answer queries. An important characteristic of such a domain is that actions may be mutually exclusive or causally related. Moreover, there is (partially complete) domain specific knowledge about how queries should be answered since an average user is a sophisticated domain expert. Query plans in these domains should not only answer the query but also respect any user intent or domain guidance provided to improve the perceived quality of the result and query execution time. Previously, methods like rule inversion and view unfolding have been found to be more effective than AI planning in obtaining access sequences for sources without interactions, a case when sources are just repositories of data.

We present a solution in SHQPlanner, a hierarchical temporal planner for query planning and execution monitoring in complex domains based on previous theoretical work on HTN planning with partial domain knowledge on one hand and temporal reasoning for query cost on the other. SHQPlanner is a sound, complete, and efficient domain-independent query planner that can incorporate partial query decomposition, source preferences, data and application interaction, and temporal constraints.

Introduction

The amount of genomic data available for analysis online is vast and ever growing. Yet, a biologist wishing to gain insight from them is lost in data model, data formats, and interfaces of particular data sources. Many sources have data as formatted file with specialized Graphical User Interfaces (GUIs) and retrieval packages. The design choices made by the autonomous data sources considers the complexity of data, efficiency of analytical tools, multi-platform support and cost of implementation, but not integration issues which would have lead to more adoption of database management systems. The heterogeneity of the data sources and the multitude of non-standard implementations makes providing uniform access for such data sources an integration nightmare.

The computer science field of data integration (also known as information integration or information gathering(Lambrecht & Kambhampati1996)), which lies at the cross-roads of Artificial Intelligence and Databases, studies how to provide access to multiple autonomous heterogeous data sources in a uniform fashion(Levy1998). Given a global world model, a set of information sources, a mapping of contents of information sources to the world model, and a query on the world model, the objective is to return information contained in the information sources that answers the query on the world model. An agent integrating the different heterogeneous sources must return only the actual information that satisfies the user's query and no more.

It turns out that biology[1] is both complex and large compared to previously considered domains in data integration. The specific characteristics are:

- In bioinformatics, data from sources and specialized applications (either located at the sources or implemented by the mediator) must be combined to answer queries. For example, gene expression data from biochips is clustered by a suitable application or a search for proteins should be fed to the BLAST application for protein similarity search and only similar proteins, which is the result of the application, should be used for pathway analysis.
- The user, a biologist, is a specialist of the domain and has (partially complete) domain specific knowledge about how queries could be efficiently answered for meaningful insights. A form of domain knowledge, the search control knowledge, may dictate that subgoals of actions may be mutually exclusive or causally related.
- Additionally, since there is considerable choice for sources and applications (pre-processing, post-processing or analytical) of a particular type(Baxevanis2002) (e.g., protein, pathway, publication, gene expression data), the user usually has strong preferences about which sources and applications are used to answer queries.
- Query decomposition in bioinformatics has a mixed-initiative planning flavour. The reason is that queries in biology can take very long time due to extreme range of data sizes that may be retrieved. Hence, biologists quickly want to decide if a line of biological exploration is worthwhile before investing more time in it looking for refined results.

[1] We are particularly interested in Bioinformatics, which is the application of information sciences (mathematics, statistics and computer science) to increase our understanding of biology.

Conventional data integration approaches consider the data sources as repositories of data but not as applications (which may in turn embody complex interactions with other sources), and they do not provide any mechanism for leveraging domain-specific user guidance. *We argue that data integration in domains with these characteristics are best addressed by using AI planning for query decomposition.*

AI Planning has been considered in conventional data integration to determine the best way to integrate data from different sources(Knoblock1995; Knoblock & Ambite1998; Kwok & Weld1996), and monitor the actual execution of source requests. Planning tackles the problem of composing a sequence of actions so that an agent can go from the initial state to the goal state given the set of legal executable actions(Kambhampati & Srivastava1995). However, planning for query decomposition seems to have lost support in favour of cheaper methods (Levy1998) like rule inversion(Duschka1996) and view unfolding(Qian1996). The main reason is that since the search space is made up of *information* states, there is no subgoal interaction among actions as they can always be executed on the sources to gain the information needed. Hence, the conclusion drawn was that using planning for just sequencing source-call actions is an over kill. But when sources can also be applications, they may encode physics of their interactions[2] which may prevent an action from being always applicable. For example, if a user's authentication request is denied (by a trusted third-party), her already available credentials (information like password or certificate) to access a source may become invalid. Hence, we need to revisit planning for data integration to address action interactions.

Hierarchical Task Network (HTN) Planning (Erol1995) is a paradigm in planning to capture user intent about desirable solutions and what actions are used in them. However, duration (temporal properties) of an action has not been widely modeled in HTN. In this paper, we extend a recent forward chaining (also called forward state-space) temporal planner, *Sapa*(Do & Kambhampati2001) to handle task hierarchies (also called *schemas* in HTN planning) and other features useful for biology. Specifically, we describe *SHQPlanner*, a hierarchical temporal planner, for query planning and execution monitoring in biology and other such domains where data from sources and applications have to be integrated, and user has background knowledge about what kind of plans are acceptable. The advantage of our approach is that the hierarchy can embody the domain knowledge while the temporal specification can help reasoning with query cost models. Important features of *SHQPlanner* are: (a) it has the ability to incorporate partial query decomposition and source preferences, data and application interaction, and incremental updates (b) it is a domain-independent query planning algorithm (c) it reasons with temporal constraints for cost estimates and (d) it is sound, complete, and efficient.

Our work is in the context of an end-to-end XML-based Bioinformatics application called *e2e*(Adak et al2002) that facilitates analysis of gene expression data by providing a uniform access to diverse online data sources (i.e., proteins, literature, pathways and gene expression data) and representing the annotations on the intermediate data in a XML format, eXpressML(Adak et al2001). For example, text summarization can be performed on the result from a literature source and the top few keywords are represented in eXpressML. In *e2e*, eXpressML can be queried by a XML language or processed by advanced statistical tools. We are in the process of integrating *SHQPlanner* into *e2e*.

Here is the outline of the paper: we describe the bioinformatics domain in the next section and survey the existing methods here for data integration. Next, we pose query decomposition as a planning problem and show how *Sapa*, a temporal planner, can be extended to support schemas. We then describe the working of the new planner, *SHQPlanner*, and present initial results. Finally, we conclude with pointers to future work.

[2]The fact that some predicates are not true as the result of an action but not others, is governed by the physical knowledge about the world being modeled.

Bioinformatics: A complex data integration domain

We are interested in data integration in the large and complex domain of bioinformatics to facilitate data analysis. Here, according to a recent survey(Baxevanis2002), there are atleast 335 data sources in early 2002[3] with at least 6-10 sources of similar type (for example, protein, pathway, publication, gene expression data, etc.). In contrast, conventional data integration solutions have dealt with very few sources with little overlap in content. Moreover, there is rich domain information on how results for queries should be obtained and strong user preference for sources (example, one biologist may prefer SWISS-PROT to PIR for protein information due to its affiliation). The completeness (but not correctness) of the results is negotiable in favor of performance and timeliness. The data size can be large (in megabytes or gigabytes). Finally, the user may want to employ the unique native data analysis capabilities of the data sources.

Data analysis in bioinformatics is done in two phases:

1. Identify genes which constitute an interesting regulatory pattern by applying a set of statistical analysis/ clustering methods like hierarchical, k-means, fuzzy and self-organizing feature maps.
2. Identify functional relationships among selected genes based on maximum number of information/ data sources to develop and verify hypotheses. In a sense, the clusters from the first step are characterized by a set of meaningful features from the databases. Additionally, relationship among the genes is predicted and validated/understood.

Karp (Karp1996) has identified the issues in data integration in bioinformatics and alluded to the need of a planning module through what he calls a Relevance Module. An example of a query that the biologist may want to ask in *e2e* is (modified slightly from (Karp1996)):

Q: *Find examples of co-expressed genes that code for enzymes in a single metabolic pathway.*

[3]Up from 281 in 2001.

Query Q can be more precisely described as - find a set of genes G from a pathway P where:

- There exists R, the set of all reactions in P
- There exists E, the set of all enzymes that catalyze a reaction in R
- G is the set of genes that encode an enzyme in E, and
- Genes in G are similar in their expression level according to some algorithm.

To answer the query, the system needs to access a protein pathway database like KEGG, the user's gene expression data and a pathway similarity algorithm. We generate the necessary query plan using *SHQPlanner* at the end.

Existing Approaches

A data integration system can be characterized by the amount of transparency it provides to the user. The different types of transparency, in increasing degree of abstraction are:

1. Format transparency or the user not having to know about the data formats supported by a source and the individual ways to access them.
2. Location Transparency or the user not having to know about the location of a piece of information on the source. This gives the impression of a single location for all information while the access details to the source is handled by the system.
3. Schema Transparency or the user not having to know the schemas of individual sources to reconcile the final result. This allows the user to query data across sources in an integrated manner.
4. Source Transparency or the user not having to know if a particular source exists. For example, the user will know that protein information can be obtained from the system but she will not have to know the source from which such an information can be obtained. Source transparency necessitates a domain ontology(Benjamin et al1998; Critchlow et al1998) so that the user interacts with the system using domain concepts and the concepts are in turn reconciled with available sources.

A variety of approaches have been developed for integrated access to heterogeneous data sources in genomics. In the *link-driven federation* approach (e.g., SRS(Etzold & Argos1993), LinkDB(Fujibuchi et al2001)), the user can browse the content of a source and also switch sources using system-provided hyperlinks. These systems provide limited format transparency to the user. In *view integration* (e.g., DiscoveryLink(Haas et al2001), K2/Kleisli(Davidson et al2001)), a virtual global schema in a common data model is created using the source description. Queries on the common data model are then automatically decomposed to source level queries. A variation of view integration is the *warehousing approach* (e.g., GUS(Davidson et al2001), (Widom1995)) where instantiation of the global schema is created, i.e., all data of interest is locally stored and maintained for integrated access. Both view and data integration provide format, location and schema transparency to the user. The holy-grail in data integration is to provide source transparency as well, which leads to *semantic integration*. In the TAMBIS(Goble et al2001) system, a common ontology of molecular biology describes the concepts of interest and data source characteristics are directly mapped to the common concepts. The user interacts in the ontological realm while the system deals with the sources.

While transparency is a desirable property to have in genomics, users may selectively want to lose it to gain control over how the queries are answered and utilize source specific features. The drawback of the existing systems is that while they handle transparency for the user, they do not model the relationships between the applications and the data sources in a formal and general framework. Consequently, even *if available*, they fail to leverage a biologist's (user's) background knowledge about how a query should be answered for her purpose or respect user's intent in the final plan. Since a typical data integration system is interactive, it makes sense to employ additional inputs from the user about query optimizations or preferences of data sources.

A Hierarchical Temporal Planning Solution

The requirements for gathering information in bioinformatics domain lead us to look back at AI Planning for query decomposition[4]. A planning problem P is a 3-tuple $\langle I, G, A\rangle$ where I is the complete description of the initial state, G is the partial description of the goal state, and A is the set of executable (primitive) actions. An action sequence S is a solution to P if S can be executed from I and the resulting state of the world contains G.

A HTN planning problem(Kambhampati et al1998) can be seen as a planning problem where in addition to the primitive actions, the domain contains schemas which represent non-primitive (abstract and non-executable) actions and acceptable rules to reduce non-primitive actions to primitive and other non-primitive actions (hence an hierarchy of actions). All non-primitive actions are eventually reduced to primitive actions so that the resultant plan is executable. The acceptable solutions to a HTN problem not only achieve the top-level goals but can also be parsed in terms of the non-primitive actions that are provided for the top-level goals(Barrett & Weld1994).

Sapa(Do & Kambhampati2001) is a heuristic forward chaining (also known as forward state space) planner that can handle actions with durations, metric resource constraints and temporal deadlines. It starts from the initial state and applies actions as they become applicable taking into account their duration and when (either start or end of the duration) each effect becomes valid. In order to guide its search, Sapa builds a temporal relaxed planning graph, and uses action and resource measures to calculate heuristic distance to the goal.

We present *SHQPlanner* in which *Sapa* is extended by introducing non-primitive actions into the domain actions, A, and providing reduction schemas for them. Moreover, we

[4]An altogether different reason to consider planning in data integration is execution monitoring(Knoblock & Ambite1998) but we do not focus on this here.

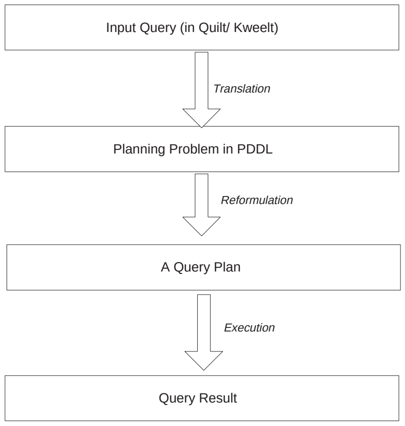

Figure 1: *Stages in processing the XML Kweelt/Quilt query of e2e with SHQPlanner.*

allow users to give preferences over parameter values of actions and schemas. We leverage the duration modeling of actions so that we can reason about query costs. We also modify the heuristic estimates to account for merged actions in the partial plan.

An XML Kweelt(Sahuguet2001) query in *e2e* goes through the stages described in Figure 1. Each query posed by the user corresponds to a new query planning problem, P_i where the initial states I and G states are different but it uses the same domain actions, A. The solution to P_i is executed by the query executor to obtain the final result. In case of failure, the query executor will pose a different planning problem $P_i^{'}$ based on failure and any partial result it already has accumulated.

Schema Specification

The specification of schema in *SHQPlanner* is allowed through the $:schema$ construct which is described in Figure 2. In this, the $:duration$ field records the minimum and maximum duration of each sequence of actions which are permissible while reducing this schema. The $:precondition$ field is a place holder to specify additional preconditions beyond those of the constituent primitive actions which should be true to apply the merged action. The $:effect$ field records the *primary effects*(Kambhampati et al1998) of the schema for which the merged action should be introduced into the plan. This takes care of basic concern in HTN planning that a very complex plan is considered because a non-primitive action is used to achieve secondary effects. The $:method$ field specifies the sequence of actions that are to be used for reductions. If there is more than one sequence, a *choice* delimiter is used.

As an example, in Figure 3, a schema is given which encodes that the primary reason to make use of this schema is to prepare the data. In order to achieve the effect, there is a choice between three sequences of actions corresponding to the alternative reductions available.

```
(:schema ⟨schema-name⟩
    :parameters
        ({⟨?var⟩ - ⟨var-type⟩})
    ;; Static duration
    :duration
        (⟨min-dur⟩, ⟨max-dur⟩)
    ;; Preconditions true at the start
    :precondition
        [(and] {⟨predicate⟩ - (⟨start-time⟩, ⟨end-time⟩)}[)]
    :effect
        [(and] ⟨predicate⟩) - ⟨effect-time⟩ [)]
    :method
        [(choice] [{(sequence] { ⟨action⟩ - ⟨duration⟩ } [)] } [)] )
```

Figure 2: *The format for schema specification. Fields in [ ] are optional while fields in { } are one or more.*

```
(:schema RESULT-PREPARE-SCHEMA
    :parameters
        (?d - data ?q - query)
    ;; Static duration
    :duration
        (st, + st 4)
    ;; Preconditions true at the start
    :precondition
        ()
    :effect
        (prepared_data ?d) - et
    :method
        (choice
            (sequence
                (CLUSTER-DATA ?d)
                (PREPARE-DATA ?q ?d))
            (sequence
                (ALIGN-DATA ?d)
                (PREPARE-DATA ?q ?d))
            (sequence
                (SUMMARIZE-PUBLICATION ?d)
                (PREPARE-DATA ?q ?d))))
```

Figure 3: *An example schema.*

```
Algorithm: Generate-Merged-Actions
Input:     Schema $s$

Output:    $M$ = [] ; set of primitive actions
1. For each action sequence $L_i$ in $s$, create action $M_i$
2.      $M_i$.name = Make-unique-name($s$)
3.      $M_i$.parameters = $s$.parameters
4.      $M_i$.precondition = $s$.precondition
5.      $M_i$.duration = 0
6.      $M_i$.primary = $s$.effect
7.      For each action $a_j$ in $L_i$
8.          If $a_j$ is non-primitive, iterate over
            Generate-Merged-Actions($a_j$)
9.          $M_i$.duration = $M_i$.duration + $a_j$.duration
10.         If($M_i$.duration $\ni$ [$s$.min_dur, $s$.max_dur])
                continue
11.         $M_i$.parameter = $M_i$.parameter $\bigcup$ $a_j$.parameter
12.         $M_i$.precondition= $M_i$.precondition $\bigcup$ $a_j$.precondition
                          - $M_i$.effect
13.         $M_i$.effect = $M_i$.effect $\bigcup$ $a_j$.effect
14.     End-for
15.     $M$ = $M$ $\bigcup$ $M_i$
16. End-for
```

Figure 4: *Procedure to produce primitive (merged) actions based on reductions in a schema.*

Reduction of a Schema

A HTN planner can transform any non-primitive action into executable actions by recursively applying the available reduction information from the schemas. Since *Sapa* is a forward chaining planner, one can interpret the reduction of a schema eventually into a *sequence* of primitive actions. We use this insight to pre-process (specifically, top-down parse) the schema into a set of *merged actions* that correspond to the sequential execution of the primitive actions in the final reduction. Figure 4 describes a procedure to create such actions in a top-down manner. The merged actions along with the original primitive actions are fed to Sapa for its normal execution.

The merged actions are like primitive actions except that they also record primary effects which will be used during search. Also, if the duration of a sequence lies outside the range of permissible durations, no corresponding merged action is created. The merged actions can be now used by *Sapa* to plan in the regular way.

Specification of Value Preferences

A $:prefers$ construct is introduced in the domain description to allow a user to specify her preference for a variable's values. This information is used while instantiating variables of a particular type in an action. In Figure 5 for example, while considering variables of $Type_1$, $value_{11}$ is the most preferred value followed by $value_{12}$, and so on. Using this mechanism, the user can provide information to the planner like query PIR only if a query on data source SWISS-PROT has failed.

```
(:prefers ($\langle Type_1\rangle$ $\langle value_{11}\rangle$ $\langle value_{12}\rangle$ ...)
          ($\langle Type_2\rangle$ $\langle value_{21}\rangle$ $\langle value_{22}\rangle$ ...))
```

Figure 5: *Specification of value preferences.*

```
Algorithm: Heuristic-Adjust-Minimal
Input:     Partial plan, $P$
Output:    Adjusted heuristic value of the plan

1. $length$ = Sapa-heuristic($P$)
2. $solution$ = Sapa-relaxed-plan($P$)
3. Foreach merged action, $M_i$ in the
$solution$ of relaxed problem
4.     Foreach action, $a_j$ constituting
       the merged action
5.         If $a_j \in solution$
6.             numMergeRedundantActions ++
7.         End-if
8.     End-for
9. End-for
10. $length$ = $length$ + numMergeRedundantActions
```

Figure 6: *Heuristic adjustment for potential non-minimal plans.*

Heuristic Adjustments

Sapa uses heuristics based on actions and resource usage to guide its search. With merged actions also added to the set of original primitive actions, we have to account for the fact that the merged actions signify user intent. We ensure this by increasing the heuristic estimate of a plan by some ϵ for each effect supported by a primitive action in place of an available merged action.

We also want to discourage plans where primitive as well as the merged actions are present to provide the same effect because the plan could possibly be non-minimal. In Figure 6, a procedure is described to capture this requirement by increasing the heuristic value of such a plan by the number of potentially redundant primitive actions.

Soundness and Completeness

A planner is sound if it generates executable plans and it is complete if the planner will find a solution whenever one exists. Since *Sapa* is both sound and complete(Do & Kambhampati2001), the only complication is introduced by the merged actions that are produced as a result of schema reductions.

In *SHQPlanner*, as part of the schema elicitation process, a verification procedure ensures that (a) all the primitive actions mentioned in the schema are declared, and (b) each action sequence in the reductions lead to some executable sequence of primitive actions. Therefore, the merged actions are ensured to be executable and consequently, any plan produced by *SHQPlanner* is sound.

Completeness is guaranteed in *SHQPlanner* as long as no plan without the merged actions are pruned away. We never

```
⟨A0⟩. 0.0 – RETRIEVE-DATA (stanford geneexp)
             :duration 2.0

⟨A1⟩. 0.0 – RETRIEVE-DATA (kegg pathway)
             :duration 2.0

⟨A2⟩. 2.0 – EXTRACT-GENES-ENCODING-DATA (pathway)
             :duration 1.0

⟨A3⟩. 2.0 – CLUSTER-DATA (geneexp)
             :duration 1.0

⟨A4⟩. 3.0 – FIND-COXPRESS-GENES (pathway geneexp)
             :duration 3.0

⟨A5⟩. 6.0 – PREPARE-DATA (q1 pathway)
             :duration 3.0
```

Figure 7: *Query plan for the example query Q.*

```
⟨A0⟩. 0.0 – RETRIEVE-DATA (pir protein )
             :duration 2.0

⟨A1⟩. 0.0 – RETRIEVE-DATA (swiss-prot protein )
             :duration 2.0

⟨A2⟩. 2.0 – ALIGN-DATA (protein )
             :duration 1.0

⟨A3⟩. 3.0 – PREPARE-DATA (q1 protein )
             :duration 3.0

⟨A4⟩. 6.0 – VISUALIZE-RESULT (q1 protein )
             :duration 1.0
```

Figure 8: *A query plan without schemas.*

prune such a plan but only penalize it through heuristic adjustments to be further down in the search queue.

Hence, *SHQPlanner* is both sound and complete.

Initial Results

We have incorporated the discussed extensions in *Sapa* (and a few search pruning tricks known to work well in forward chaining algorithms) to build the *SHQPlanner.* Additionally, we have built a small domain describing the actions to query gene expression, protein, publication and pathway sources, and run clustering, sequence alignment and text summarization applications for bioinformatics.

Running *SHQPlanner* as a traditional query planner, we show the query plan returned for the example query, Q. $Stanford$ refers to a public gene expression database while $KEGG$ is a pathway database. Two data sources and a standard clustering application are needed to complete the query. Thus, the query planner does not have to depend on user input for generating a valid query plan.

```
⟨A0⟩. 0.0 – MERGED:DATA-FETCH-SCHEMA:
             %RETRIEVE-DATA%ALIGN-DATA
             (pir protein )
             :duration 3.0

⟨A1⟩. 0.0 – MERGED:DATA-FETCH-SCHEMA:
             %RETRIEVE-DATA%ALIGN-DATA
             (swiss-prot protein )
             :duration 3.0

⟨A2⟩. 3.0 – PREPARE-DATA (q1 protein )
             :duration 3.0

⟨A3⟩. 6.0 – VISUALIZE-RESULT (q1 protein )
             :duration 1.0
```

Figure 9: *A query plan with schemas. The merged actions will be replaced with the corresponding action sequences in a post-processing step.*

Now, suppose the biologist wants to retrieve aligned protein data and see the matched sequences in a viewer. In Figure 8, a query plan is shown for the problem where data is retrieved from the two protein sources and then they are aligned with each other. But if the user only wants to align proteins of each source respectively, it is much simpler to realize the requirement with a suitable schema(*DATA-FETCH-SCHEMA* here). Figure 9 has the resultant plan.

Running the query planner in the bioinformatics domain for a number of user queries has shown that the end users are generally favorable to specifying schemas in order to control the resulting query plan. The main challenge currently is to provide a suitable user interface so that the biologist is free from syntactic hassles. The query plan is generated in a few milliseconds.

Query decomposition as mixed-initiative planning: Queries in biology can take very long time due to sheer size of data to be retrieved. The domain is also characterized by high variability in data sizes – some attributes are in a few hundred bytes while others are in megabytes. The ability of *SHQPlanner* to reason with action durations is very useful for the biologist in exploring alternative query plans based on changing deadlines. Specifically, they want to quickly decide if a line of biological exploration is worthwhile before investing more time in it looking for refined results. In future, we propose to use runtime statistics from the query executor to update duration information.

Conclusion and Future Directions

In this paper, we considered data integration in a complex domain (bioinformatics) where sources can be data repositories as well as applications, and presented a hierarchical temporal planner to perform query. Another characteristics of the domain - bioinformatics - is that the user is a biologist who has specialized knowledge about the domain. Specifically, the user has (partial) schemas or domain knowledge about how a query should be resolved, e.g., a solution for protein search may be to fetch data, merge results, run a particular application, and show the result. Prior work in hierarchical task network (HTN) planning has approached such problems with partial schemas and actions. The approach considered non-primitive actions and their reduction schemas as part of the domain specification (i.e., the set of available actions) and generalized the usual refinements to handle non-primitive actions. We used similar techniques in *Sapa* to capture user intent. Additionally, we used Sapa's temporal engine to model query costs and prune long plans.

In future, we plan to extend the work in many directions. First, the role of planning can be extended to query execution by allowing for plan monitoring and replanning in the event of failure(Knoblock & Ambite1998). Second, we have to make the bioinformatics domain richer with more primitive actions for additional types of sources. Third, a convenient user interface must be developed to make schema elicitation from the biologist easier. Finally, since *SHQPlanner* itself is a domain independent planner, we want to investigate its usage in more conventional temporal planning domains.

Acknowledgements

I wish to thank Minh Binh Do for providing the *Sapa* code, Sudeshna Adak for useful discussions about the bioinformatics domain, and Subbarao Kambhampati for pointers on planning approaches for data integration and comments on an earlier draft of this paper.

References

Adak, S., Srivastava, B., Kankar, P., and Kurhekar, M. 2001. A Common Data Representation for Organizing and Managing Annotations of Biochip Expression Data. *Unpublished Technical Report.*

Adak, S., Batra, V., Bhardwaj, D., Kamesam, P., Kankar, P., Kurhekar, M., and Srivastava, B. 2002. Bioinformatics for Microarrays. *Submitted for publication.*

Barrett, A., and Weld, D. 1994. Task Decomposition via Plan Parsing. *Proc. AAAI.*

Baxevanis, A. 2001. The Molecular Biology Database Collection: 2002 update. *Numcleic Acids Research, Vol. 30, No. 1*.

Benjamin. V.R., Fensel, D., and Perze, A.G. 1998. Knowledge Management through Ontologies. *Proc. 2nd Int. Conf. PAKM.*

Critchlow, T., Ganesh, M., and Musnick, R. 1998. Automatic Generation of Warehouse Mediators Using an Ontology Engine. *Proc. 5th KRDB Workshop.*

Davidson, S., Crabtree, J., Brunk, B., Schug, J., Tannen, V., Overton, C., and Stoeckert, C. 2001. *K2/Kleisli and GUS: Experiments in Integrated Access to Genomic Data Sources.* IBM Systems Journal, March.

Do, B., and Kambhampati, S. 2001. *Sapa: A Domain-Independent Heuristic Metric Temporal Planner.* Proc. European Conference on Planning.

Duschka, O. 1997. Query Optimization Using Local Completeness. *Proc. AAAI*.

Erol, K. 1995. *Hierarchical task network planning: Formalization, Analysis, and Implementation.* Ph.D. thesis, Dept. of Computer Science, Univ. of Maryland, College Park, USA.

Etzold, T., and Argos, P. 1993. *SRS: An Indexing and Retrieval Tool for Flat File Data Libraries.* Computer Application of Biosciences, 9:49-57.

Fujibuchi, W., Goto, S., Migimatsu, H., Uchiyama, I., Ogiwara, A., Akiyama, Y., and Kanehisa, M. 1998. *DBGET/LinkDB: an Integrated Database Retrieval System.* Pacific Sym. Biocomputing, pp 683-694.

Goble, C., Stevens, R., Ng, G., Bechhofer, S., Paton, N., Baker, P., Peim, M., and Brass, A. 2001. *Transparent Access to Multiple Bioinformatics Information Sources.* IBM Systems Journal, Vol. 40, No.2, pp 532-551.

Haas, L., Schwarz, P., Kodali, P., Kotlar, E., Rice, J., and Swope, W. *DiscoveryLink: A system for integrated access to life sciences data sources.* IBM Systems Journal, Volume 40, Number 2.

Kambhampati, S., Mali, A., and Srivastava, B. 1998. Hybrid planning for partially hierarchical domains. *Proc. AAAI.*

Kambhampati, S., and Srivastava, B. 1995. Universal Classical Planner: An algorithm for unifying state space and plan space approaches. *In New Trend in AI Planning: EWSP 95, IOS Press.*

Karp, P. 1996. A Strategy for Database Interoperation. *Journal of Computational Biology*.

Knoblock, C. 1995. Planning, Executing, Sensing and Replanning for Information Integration. *Proc. IJCAI-95.*

Levy, A. 1998. Combining Artificial Intelligence and Databases for Data Integration. *At http://citeseer.nj.nec.com*

Knoblock, C., and Ambite, J. 1997. Agents for Information Gathering. *Software Agents, J. Bradshaw (ed), AAAI/MIT Press, Menlo Park, CA; Also at http://citeseer.nj.nec.com/169819.html*.

Kwok, C., and Weld, D. 1996. Planning to Gather Information. *13th AAAI National Conf. on Artificial Intelligence, AAAI/ MIT Press, Portland, Oregon, 32–39.*

Lambrecht, E., and Kambhampati, S. 1996. Planning for Information Integration: A Tutorial Survey. *ASU-CSE-TR 96-017.*

Pottinger, R., and Levy, A. 2000. A Scalable Algorithm for Answering Queries using Views. *Proc. 26th VLDB.*

Qian, X. 1996. Query Folding. *12th Int. Conference on Data Engineering, New Orleans, Louisiana, 48–55.*

Sahuguet, A. 2001. Kweelt. *http://db.cis.upenn.edu/Kweelt/.*

Widom, J. 1995. Research Problems in Data Warehousing. *Proc. 4th CIKM.*