The Planning Web in Action — Working Paper (Draft)

Christoph Wernhard
Persist AG
wernhard@persistag.com

November 10, 2000

Contents

1 Introduction

We propose resource oriented inference as one way of bringing the Semantic Web into action. It provides a framework for expressing and processing a variety of tasks from areas such as planning, scheduling, manufacturing resource planning, product data management, configuration management, workflow management and simulation. Resource oriented inference as a part of the Semantic Web should allow such tasks to be performed within the scope of the World Wide Web.

A prototypical application is be the purchase of a complex product with the help of the Web. The product consists of multiple parts, some of them complex products by themselves. Several services are required to compose the product. Subparts and services can be provided by different companies all over the world. Delivery time and cost of the product should be optimized.

2 Resource Oriented Inference

2.1 Basic Concepts

- *Object* — Represented by a term. Such an object term can be just a symbol, expressing the object identity, a literal (e.g. string, number) or a structured term. A structured term can be used to represent the structure of an object and also to represent an object whose identity is derived from other objects' identities (Skolemization, []).

- *Fact* — An atomic proposition.

- *Resource* — A fact that can be consumed and produced by rule application (state transition). A state can be considered as a multiset of facts.

- *Logical Fact* — A fact that is not a resource, i.e. is not consumed by rule application.

- *Rule* — Describes a possible transition from one state to a successor state. It consists of two propositions: a *head* and a *body*.

  We consider only rules in Horn-bundle form here: both head and body consist of a multiset of atomic facts. The meaning of such a rule is, that a state in which the facts of the body are present has a successor

state, in which the facts of the body are consumed (for each element of the body, a corresponding fact is removed from the state) and the facts of the head are produced (added to the state).

To express that a fact must be present, but should not be consumed by the rule application, that fact must appear in both body and head.

Logical facts in rule bodies can express constraints.

Logical facts that have been asserted for a planning task (e.g. *clear(table)* in a blocks world example) hold in all states, i.e. can be "consumed" arbitrary many times by rule applications.

- *Action* — A single action corresponds to one or more rules. An action is denoted by a functor and parameters. Parameters are variables that can be shared with variables in the rules, and thus be bound to objects during the planning process.

- *Plan* — A partially ordered set of actions. The partial ordering represents constraints on the ordering in which the actions can be executed.

2.2 Tasks and Corresponding Inference Modes

The general outset is a set of facts *START* representing a start state, a set of facts *GOAL* representing a goal state, a set of rules *RULES* and the set of plans *PLANS*, which lead from *START* to *GOAL* by state transitions according to *RULES*.

Different modes of inferencing are characterized by different instantiations of these parameters.

Planning *GOAL* and *RULES* are given. Computed is the set of plans, or a single plan, which might be preferable in some sense, or an enumeration of plans, possibly ordered according to preference criteria.

Projection *START* and a plan are given. The outcome state of the plan, when applied to *START* is computed.

Postdiction (Abduction) *GOAL* and *RULES* (and maybe a subset of *START*) are given. Computed are missing *START* facts, that would be required to reach *GOAL*.

Scheduling Similar to planning. The focus is on assignment of temporal information to actions. (Plans might be given?)

Plan Execution Some kinds of actions can be directly executed electronically, for example making an order by e-mail or document routing tasks. Depending on the application, such an execution phase might follow a planning phase or might interleaved with the planning ("Reactive Planning").

3 Product Composition

3.1 Components as Resources

The composition of a product from parts can be viewed as a transition from a state in which the parts are *available*, to a state in which the composite product is *available* and the parts are no longer so.

$$available(bike(wheel(X), wheel(Y), frame(Z))) \Longleftarrow$$
$$available(wheel(X)), available(wheel(Y)), available(frame(Z)).$$

Services needed to compose products might be explicitly represented in the same way as products.

$$available(bike(wheel(X), wheel(Y), frame(Z))) \Longleftarrow$$
$$available(wheel(X)), available(wheel(Y)), available(frame(Z)),$$
$$available(bike\_assembly).$$

4 Architecture Outline

4.1 Document Types

- ***SCHEMA*** — Describes a type (schema, class). The type system allows to declare a subtype, referring to the supertype at another URI. Notion of structural equivalence, intersection types. Functional "expression" oriented Web language view: type language expression (data term) is composed from subexpressions in different hyperlinked documents.

- ***RULE*** — Rules describing actions. Object variables in the actions are typed according to SCHEMA. Product composition and the effects of services are represented by rules.

- ***ONTOLOGY*** — Perhaps just sets of type names suffice. Such sets can be given either given explicitly or implicitly (e.g. a prefix, XML-namespace). Maybe also action names would be required. Maybe names for rule-sets are also useful? So far rules are only related to types by the typing of their parameters.

- ***OFFER*** – Product offer. Publication of an OFFER states that its author offers a product under certain terms. Technically an OFFER is a special kind of rule, like: a product of a certain kind (reference to a SCHEMA type, but possibly with some instantiated property values) is provided if a certain amount of money can be consumed by the author as actor.
- ***REQUEST*** – Product request. While requests can be "temporarily" expressed in a query, they could also be published as documents. (Also a "temporary" offer seems possible). Technically a REQUEST is a special kind of rule.
- ***PLANS*** – Would that be useful? What about complex rules?

4.2 Client View

This subsection outlines a possible interaction with the Planning Web from the client (user) side.

1. Find the relevant schemas. This must be really easy for the user: keyword search, browsing, information about a schema's authority and importance.
2. Find out what can be done with an object of a certain type. Rules for the type have to be gathered. The effect of rules might be shown by "abstract" inferences without concrete start and goal, which perhaps might be performed stepwise interactively in a browser.
3. Find related classes. Through signature of structure and rules.
4. Create a document with the query, i.e. start and goal facts and inference mode specification.
5. Run the gatherer. This is a "rough" search engine that gathers relevant documents from the web by symbol indexing (AltaVista).
6. Run the inference engine. The inference task is compiled from the query and gathered documents.
7. Process the output. Depending on the inference mode, the output might e.g. be an ordered enumeration of plans. These might be inspected manually (e.g. visualized as DAG, maybe abstracted to properties like cost, time, number of involved actors). If a plan contains actions to be executed electronically, the executor might be called.

Some combinations of steps could be run interleaved (e.g. gathering and schema/rule browsing, gathering and inferencing, inferencing and executing).

5 Notes

Query for a Product Instance The answer is an enumeration of plans with parts by different offerers, dates at which parts/services are needed. The enumeration is sorted by e.g. time/cost/number of offerers/kinds of offerers (e.g. local preferred).

Query about the Composition of a Product Such a query could be more or less general, e.g. *journey* or *journey from Germany to the USA*.

It can in part be answered from SCHEMA and RULES alone, or include (abstracted) information from offerers (e.g. price ranges).

An ONTOLOGY is specified to indicate desired level of abstraction.

Query for Possible Uses of a Given Product Instance Corresponds to prediction. Maybe the range of uses has to be restricted for such a query (goal predicate and parameter type, number of actions to be applied, cost).

Query for Missing Facts to Achieve a Goal Might be necessary to specify the range of missing facts (predicate, parameter type).

Query for a Customer for a Given Product A customer not explicitly requesting the given product, but just one having it as subpart, will be also be found.

This kind of query can be used to find out information about the demand for a particular kind of product.

Standardized Products Schemas for standardized parts — it should be possible to get symbolic identifiers (i.e. unambiguous) (maybe parameterized e.g. size, quality features?).

Descriptions of Product Composition The composition of products is more or less detailed described (e.g. Debian package system — description is rather complete, package dependency, disk-space used etc.).

Interfaces Parts might have interfaces (hardware slots, plugs, screws) of certain kinds (type). These interfaces might be free or occupied. They might have a fixed or varying number of such interfaces. The number of interfaces might be extended (plug in a card that provides more ttySs).

Different Levels of Detail Rules may be on different levels of detail: simply list a bunch of parts or describe detailed assembly.

Are there relationships among rule sets that express "compatibility" of different such levels?

Different Levels of Abstraction — Physical, Functionality Different rule sets may represent different views: e.g. composition of physical parts and abstract functionality (e.g. a *plastic_case* provides *protection*).

Knowledge about the "Usual" Some configurations are equivalent, but one is more usual (default).

Prices Might Depend on Some Factors Prices might depend on date (short-term delivery more expensive) and on amount.

Offerings/Requests Might Relate to Actual Dates

Structure Driven Configuration For some configuration tasks: Object structure as "driver"? A set of properties that must be instantiated. Express as rules?

$$configured(A, B, C) \Longleftarrow$$
$$is\_set("a:", A), is\_set("b:", B), is\_set("c:", C).$$

Offering Extras An offerer might offer additional free products, with a requested product (e.g. warranty, but also other extras). These would be considered in the plan comparison. (This is implied by the Horn-bundle rule format, in which a rule can have several consequents.)

Plan Execution A plan can in part be executed by computer (e.g. electronic orderings). Additional facts obtained afterwards could be stored (at the user's side). Re-plan with same goal but the updated "user fact set" to get a plan with the things still to do.

Two Abstraction levels:

1. Composition — *get_product.*
2. Detailed actions — *send_order, receive_product, pay_bill* (at this level automatic execution is possible).

Suspended User Interaction If user decisions are involved to "complete" a plan, they might be suspended as far as possible, i.e. the engine works automatically and afterwards enter the user dialog (like compiler error messages).

Plan Execution Monitoring Which types of documents are required to allow this? Plans, facts about what has already been done? In some applications information about plan execution should be made accessible, in other kept secret.

6 Base Library

6.1 Base Types

Means to represent these object should be in a base library and perhaps be specially supported for efficiency:

- Actor — Responsible for executing an action. (Also for making an offer or stating a request.)
- Time — absolute, time-zones, intervals, working-hours, holidays.
- Money — currencies.
- Space — cities, distances.
- Countries — taxes, customs, social/economical/political/natural facts.
- Basic Business Processes — order, reservation, warranty. Some of these can be executed electronically. (Interface to existing "standards", protocols?)
- Units of Measurements — physics, clothes.
- Person — Address. (How does this relate to actor? Does an actor have to be a person?)

6.2 Base Predicates

- *available(Object)* (product composition).
- *owns(Actor, Object)* (this is similar to *available*, but additionally qualified by an Actor argument).

7 Implementation Issues

7.1 Planning with Horn Bundle Transition Logic

There is a multitude of approaches to "resource oriented inference", as we call it here: situation calculus, transition logic, linear logic, petri nets, pi calculus, process algebra, dynamic logic, relevance logic, Markov decision processes, etc. Unfortunately there is very few work that relates these approaches, and shows their essential features and differences.

We use here the classical planning approach (situation calculus, Horn bundle transition logic) for the following reasons:

+ Clear semantics (situation calculus or equational semantics).

+ Clear separation as well as integration of resources and static information. Only propositions (not terms) are fluents.

+ Different inferencing modes (verification, abduction etc.) can be easily specified.

+ Efficiency when used for planning tasks has been shown.

+ Clean use and propagation of parameters through logical variables.

+ Similar to logic- and constraint-programming systems. Can be coupled with existing such systems (e.g. for arithmetics and constraint handling).

− Terms must be acyclic (as in first order logic and Prolog — in contrast to arbitrary graphs which are common in object oriented systems).

− No means to express resources (effects) that MUST be consumed. (Maybe this can achieved through language extensions.)

− Still unclear what kinds of tasks can actually efficiently processed by planners.

7.2 LCB as Implementation

+ Efficiency has been shown (compares very well with respect to Planning Systems Competition 1998).

+ Easy to interface with logic- and constraint-programming system (Prolog constraint programming system e.g. Eclipse).

− The PTTP style (and missing assignment in some Prologs, in case of an implementation in Prolog) might be too rigid for some strategic and algorithmic improvements.

? Inclusion of optimization of cost and time constraints.

? Heuristic use of global constraints (which follow (how?) from axioms): e.g. “at any point of time only one block is in hand”. Which role play which kinds of such constraints in the different planning algorithms?

8 Language and Implementation Notes

General Facts Support for general facts should not be explicitly listed as fact multiset: e.g . $have(10 * \$)$, $have\_time[10:00 - 16:00]$

Universal/Existential Quantification in Bodies/Heads This is considered in the planning literature.

Related to negation as failure (finish expansion of a pattern if there is no more solution)?

Sets Integration of set oriented data (e.g. from relational databases). Object types like ODMG's *set*, *bag*, *list*?

Dual: Facts that *must* be consumed Solution to ramification problem? Can LCB extended this way? Linear logic?

Language Extensions Negation as failure?

Plans with conditional? Conditional actions?

See PDDL.

Abduction Can abduction be expressed simply by rules with *true* body?

Schema Evolution Schemas can evolve by extension (subclassing).

Rule Evolution How rules/actions can evolve? Example: "that precondition is missing", "that precondition is not needed", "this can be achieved better with a different rule".

What are useful relationships among rules in this respect: subsumption?

Combination of Plans and Actions Can rules, actions and plans be combined (e.g. rules for *load*, *drive*, *unload* to a single rule *transport*) ? (Compare functional combination, statement combination ";").

Can combined rules be used for hierarchical planning, by grouping actions into layers (ontologies — sets of names)? Planning is then restricted only to use rules from certain layers.

Should rules (actions) allowed to be themselves partially ordered complex actions like plans?

Incomplete Objects and Typing Should be able to handle incompletely specified objects, i.e. objects which have for some of the properties declared with their type no value specified.

This is a bit weird (also in RDF-Schema): in programming languages usually the association of an object with a type can bee seen as a guarantee that the object is "completely specified" with respect to the type.

Solution Output Algorithmic properties for each task (in principle and for concrete algorithms): output a single solution, enumerates a set of solutions, is the enumeration already sorted in some way, does the enumeration finish?

Where do each of the application tasks actually require nondeterminism?

Classical Planning Adequacy Problems Which of the classical "Problems" are relevant:

Atomic view of time, ramification problem, qualification problem?

9 Web Specific Issues

Heterogenous representations Different schemas, different languages — wrapping and mapping is needed but might be independent of the planning mechanism, i.e. we can assume that it has been "already done" before.

Flavors of Documents fond in the Web Static documents that never change content, documents that change content, implicit documents that are accessed via a query expression.

Dimensions of Distribution Schema, Rules, Processing.

10 Related Languages and Frameworks

PDDL *Planning Domain Definition Language*

PSL/PIF (?)

11 References

11.1 Starting Points

- http://www.ai.sri.com/~wilkins/planning.html — Planning links on homepage of David E. Wilkins.
- http://www.sics.se/isl/configuration/prodcon.html — Product Configuration Systems

11.2 Further Links

- http://www.cs.hut.fi/ pdmg/ — Product Data Management Group at Helsinki University
- http://www-is.informatik.uni-oldenburg.de/~sauer/puk/index.html — GI Planning and Configuration.
- http://cs-www.cs.yale.edu/homes/dvm/ — Drew McDermott
- http://planet.dfki.de/ — Esprit Project.
- http://www.aiim.org/wfmc/mainframe.htm – Workflow Management Coalition, Wf-XML binding.
- http://www.ai.sri.com/~wilkins/arpi/po.html — Planning Ontologies.
- http://www.pdmic.com/ — Product Data Management.
- http://www.iac.honeywell.com/Pub/Tech/CM/CMTools.html (Configuration Management FAQ — seems to concern Software configuration only.

11.3 Other Possibly Related Activities

- Shared planning and activity representation ARPI/SPAR (1996)
- DARPA: CPR/ARPA/ARPI/OWMG/KRSL/POCG (Core Plan Representation CPR)
- Open Planning Architecture (O-Plan)
- OZONE, Scheduling ontology
- PERT Networks, Critical Path Method (Scheduling), early, avg, latest time
- MIT library of business processes (Process Handbook) - legale?
- Upper Penman Ontology?
- P.Hayes: a catalog of temporal theories
- STEP, Express