Electronic News Journal on Reasoning about Actions and Change 137
Vol. 1: *137-155*, 30.11.1997
http://www.ida.liu.se/ext/etai/actions/nj/9711-2/

Protocol of the discussion during November, 1997 on

Ontologies of Actions and Change

Edited by: Erik Sandewall
Linköping University, Sweden

From: Michael Gelfond on 3.11.1997

I agree with Ray that it may be a good idea to separate "ontology" from "epistemology".

For me to specify "ontology" means to give a collection of individual objects, functions and relations which comprise our domain. The type of logical connectives used to construct sentences in this language is part of "logical system" and does not belong to the ontology. Closed World Assumptions of different sorts,etc belong to "epistemology".

We seem to differ slightly on the meaning of the frame problem. To me the frame problem is a special case of a more general problem of finding logical system suitable for representing defaults and their exceptions in a way which will insure a high degree of elaboration tolerance. (Frame axiom is a particularly difficult default since it is related to causality, representation of time, etc)

This is of course only one of several possible views but I think an important one.

I believe that the ontology of situation calculus and action languages are basically the same. Situation IS a sequence of actions, etc. Our causal models (or automata) are graphs representing trees of situations. (One superficial difference is that our actions are undefined in situation where the corresponding preconditions are not satisfied, while in Ray's case they seem to "return" the same situation).

The more important difference is that action languages stress the distinction between description languages and query languages and define entailment relation directly without appealing to first-order logic.

To turn these languages into calculi one need to decide what logical system to use to formalize reasoning about causal models defined by the corresponding action theory. My favorite choice here is declarative logic programming with answer set semantics which can be viewed as a variant of default logic.

We have now several logic programming versions of situation calculus proven to be sound and complete w.r.t. entailment in action description languages.

The disadvantage of this approach is that we need to develop more mathematics for dealing with declarative logic programs (default theories) which is a slow process. Even though there has been an important progress in this in the last few years, classical logic is still a better choice in this respect. I believe however that in a long run default logics will allow for more elaboration tolerant and computationally better representations of various types of commonsense knowledge.

This is related also to the comment by Mikhail Soutchanski when he stresses the differences between classical and non-classical logics for knowledge representation. If formalizations of different domains are all done in default logic then we will have no problem to combine all of them together. For instance, if our initial situation is defined in Datalog then it combines very nicely with logic programming version of situation calculus. Even if the languages used for encoding knowledge in different domains are special purpose (like action languages) we will not have any problem combining them together if their entailment relations are formalized in the same language. (e.g. Instead of $\mathcal{A}$ we can use its sound and complete formalization in default theories).

I am not claiming BTW that nonmonotonic logics are better than classical one. I do not take it as a truth (revealed or otherwise). Pat (and Ray?) can be right and commonsense reasoning can be compatible with monotonic logic. But I think that it is an open and difficult research question which can be answered but trying various approaches.

One more comment on how action languages are used. Suppose you have some collection of actions and their effects and would like to write a planner which examines, in some reasonable order, possible sequences of actions and checks if your domain description entails that the goal G holds after the sequence of actions is executed. In this case domain description in $\mathcal{A}$-like language together with its entailment relation can be used to precisely specify the problem.

The corresponding logic programming formalization of this entailment together with the domain description forms the main part of the planner. If you decide to use Prolog inference engine you may need to slightly transform the program to avoid floundering, etc. If you do it using results from the theory of logic programming the resulting program will be provenly correct. It seems to be very similar to what people in Toronto do, except we probably concentrate more on correctness of actual program.

There are of course many other uses.

Finally, some corrections to Ray's comment on $\mathcal{A}$-languages. They have histories (actually, it is their basic feature). Language $\mathcal{L}$ from this family combines branching and linear time. This is published in [j-jlp-31-201], which is a special issue on Logic Programming and Reasoning about actions and I think it contains good papers.

As Vladimir already mentioned there are (fully) first order languages.

References:

[j-jlp-31-201] Chitta Baral, Michael Gelfond, and Alessandro Provetti. *Representing Action: Laws, Obervations and Hypotheses.* Journal of Logic Programming, vol. 31 (1997), pp. 201-244.

From: Mikhail Soutchanski on 3.11.1997

I will follow the distinction between the gof-sitcalc (sitcalc of 1969) and the R-sitcalc (sitcalc of 1997+); this distinction is proposed by Pat Hayes in

ENRAC 31.10.1997.

> The situation calculus is a foundation for general purpose high-level programming languages. Note that it is an easy exercise to formalize the Turing machine in the SC.

It is pretty easy to formalize a Turing machine in almost anything, so this doesnt mean much. But in any case, what has this got to do with the topic we are all concerned with? How did Turing machines come to be relevant here?

My intention was to turn the attention to computational aspects: the R-sitcalc is a general theory of action that is a foundation for the high-level logic-programming language GOLOG and its descendants (mentioned by Ray Reiter, ENRAC 27.10.1997). Because a version of GOLOG is used to control a robot in the real world, the R-sitcalc (as a formalism for reasoning about actions) can be judged according to the test suggested by Murray Shanahan (ENRAC 23.10.1997).

> Most intuitive reasoning done by humans lies entirely outside the purview of the situation calculus.
>
> Note that your objection can be easily rephrased as: "Most intuitive reasoning done by humans lies entirely outside the purview of the formal logic".

I have no idea what this response is supposed to mean. Do you identify formal logic with the situation calculus? Or do you mean only that much of intuitive reasoning is outside the scope of our subject? Or what??

I do not identify formal logic with the situation calculus. But it is true that I am not sure whether formal logic corresponds precisely to much of our intuitive reasoning. My point was that it is not fair to judge the R-sitcalc only according to the criterion whether it exactly captures *our* intuitions about actions, changes and situations (by the way, nobody ever claimed that the gof-sitcalc or the R-sitcalc expresses the final truth). As long as robots can successfully execute high-level programs based on the R-sitcalc, it is not completely worthless.

> I'm not sure whether we must have the same concerns that the cognitive science has. Most of the people do not think in terms of C, LISP, PROLOG, but all these languages are still useful for writing programs that will exhibit an intended behavior. Similarly, the SC is useful as the basis for the high-level programming language.

And again, I'm at a loss to understand your point. What is 'the high-level programming language' you refer to here? For the record, I think that unless our subject is part of cognitive science, then it's just empty formula-hacking. See earlier comments in reply to Erik.

'The high-level programming language' = GOLOG. There are several experiments performed by anthropologists and psychologists with different people in different parts of the world. As far as I understand their results, in some cases, ability to solve naive physics problems or derive conclusions from syllogistic premises depends on cultural background, education and other personal factors. For this reason, psychology (and cognitive science) cannot be *the only* foundation of our research.

From: Rob Miller on 6.11.1997

Hector,

I'd like to express agreement with your first point, that KR is about modelling. But I'd like to take issue with a couple of your other points, (3) and (5). In point (3) you said:

> *The remaning problem, that we can call the* semantic *problem, involves things like the frame problem, causality, etc.*
>
> *To a large extent, I think the most basic of these problems have also been* solved*:*
>
> *Basically, thanks to Michael and Vladimir, Erik, Ray, and others we know that a rule like:*
>
> *if A, then B*
>
> *where A is a formula that refers to time i or situation s, and B is a literal that refers to the next time point of situation, is just a constraint on the possible transitions from the the states at i or s, and the following states.*
>
> *Or put in another way,* **temporal rules are nothing else but a convenient way for specifying a dynamic system (or transition function)**
>
> ......

My problem with this is that, in general, dynamical systems in the everyday world can't be realistically modelled as state transition systems, because they involve things like continuous change, actions or events with duration, partially overlapping events, interuptable events, etc. That's why other communities involved in modelling dynamical systems (e.g. physicists, engineers, the Q.R. community) choose to model time as the real numbers. In this case, there is no "next time point", so it's difficult to read "if A, then B" as a constraint in the way you suggest. The analogy between everyday dynamical systems and state transition systems/database updates only works for a relatively small class of carefully picked domains.

Your point (5) was:

> *It's not difficult to change the basic solutions to accommodate additional features (e.g., non-deterministic transition functions, unlikely initial conditions, concurrent actions, etc.) in a principled way.*

Well again, it seems to me that if this is true, it's simply because researchers tend to pick "additional features" to work on which will conveniently fit into the state transition view of the world, as opposed to picking from the rather large collection of issues that won't.

From: Ernie Davis on 6.11.1997

Ray Reiter writes, in newsletter ENRAC 23.10

> *qualitative physics and planning have no difficulty with the FP because, without exception, they adopt the STRIPS sleeping dogs strategy. Which is to say,* **they assume they have complete information about world states**.

I don't think that this is quite right in the case of qualitative physics. My KR '92 article "Axiomatizing Qualitative Physics" [c-kr-92-177] presents a theory which, being in first-order logic, is perfectly able to characterize inferences from partial information, but does not require any special frame axioms for the continuous parameters. The reason is that the behavior of a continuous parameter is governed by governed by a qualitative differential equation of the form, "The derivative of P is the sum of the influences on P". P remains absolutely constant if the sum of the influences is zero. P retains the same qualitative value to some next modes of the system if it is consistent that some other parameter should change its value before P does. In any case, the behavior of P in staying the same is governed by the same law that governs its behavior in changing. No special law is needed to cover the cases where P stays the same. (For discrete parameters, I did need a frame axiom.)

More generally, for those, like Pat and me, whose primary interest is physical reasoning, a temporal ontology whose central category is "a finite sequence of actions" seems awkward at best. Physical reasoning is chiefly concerned with continuous, asynchronous, external change, and it is much easier to deal with this by making the continuous time-line primary and adding actions on top of that, rather than vice versa.

– Ernie Davis

References:

[c-kr-92-177] Ernest Davis. *Axiomatizing Qualitative Process Theory.* Proc. International Conf on Knowledge Representation and Reasoning, 1992, pp. 177-188.

From: Hector Geffner on 10.11.1997

Rob Miller says:

> *My problem with this is that, in general, dynamical systems in the everyday world can't be realistically modelled as state transition systems, because they involve things like continuous change, actions or events with duration, partially overlapping events, interuptable events, etc. ...*

My point is that action languages - in any dress you like - are just a convenient means for specifying (and in certain cases reasoning with) dynamic systems. That is the main lesson I think of the Yale Shooting Problem(s) and a lot of the work on temporal non-mon. Namely, the meaning of a rule like:

```
if loaded and shoot then not alive
```

is that the only state trajectories s0, s1, s2, ..., that are possible are the ones in which 'alive' is false at si+1 when loaded and shoot are true at si.

You can formulate the idea in many ways (suitable circumscriptive policy, Erik's version of chronological minimization, predicate completion, ...), but it is the same idea: *rules specify possible state transitions, observations prune possible state trajectories.*

Now, Rob is right; dynamic systems come in different varieties; e.g.,

1. discrete time, discrete value space
2. discrete time, continuous value space
3. continuous time, continuous value space
4. ....

Rules like the one above (with first order extensions, etc) are good for specifying systems of Type 1 only. Yet it's not difficult to see how systems of Type 2 could be specified as well.

Actually there are *other* type of mathematical models for the type of problems that Rob has in mind as the "Semi-Markov Decision Processes" (probabilistic continuous processes - like queuing systems - that are controlled at discrete time intervals).

My point is that we are *not* inventing new mathematical models of dynamic systems. What we are inventing are suitable structured languages for specifying and in certain cases controlling those systems. That's what STRIPS is about.

In my view, the KR/control enterprise is about developing richer versions of STRIPS suitable for specifying and controlling not only systems of Type 1, but also Markov Decision Processes, Partially Observable MDPs, Semi-MDP's, etc etc.

How we will measure success?

When we can model and control some dynamic systems that cannot even be modeled using non KR methods.

- Hector Geffner

From: Rob Miller on 11.11.1997

Hector Geffner (ENRAC 10.11) wrote:

> *Now, Rob is right; dynamic systems come in different varieties; e.g.,*
>
> 1. *discrete time, discrete value space*
> 2. *discrete time, continuous value space*
> 3. *continuous time, continuous value space*
> 4. *....*
>
> *Rules like the one above (with first order extensions, etc) are good for specifying systems of Type 1 only. Yet it's not difficult to see how systems of Type 2 could be specified as well.*
>
> *Actually there are* other *type of mathematical models for the type of problems that Rob has in mind as the "Semi-Markov Decision Processes" (probabilistic continuous processes - like queuing systems - that are controlled at discrete time intervals).*

That's right. But I think that an important wider problem that we have to tackle within "reasoning about action and change" is how to synthesise or combine very different approaches to modelling dynamic systems within a single "commonsense" framework. For example, I'd like to see more research along the lines of Erik Sandewall's 1989 work on combining reasoning about actions with modelling using the differential calculus. It's true that there has been a small amount of subsequent work on this theme since then (see e.g. `http://www.dcs.qmw.ac.uk/~rsm/project.html#Other` for a bibliography). But not much compared with, say, work on extending state-transition based approaches to deal with ramifications in evermore sophisticated ways. Why is this so? Why don't we put much effort into addressing challenges such as Kuipers' - on combining the Situation Calculus with Q.R. (see Kuipers' book, p. 201)? If we did more of this type of work, we'd stand more chance of being able to (in Hector's words) "package the theory for the outside world".

Rob

From: Erik Sandewall on 13.11.1997

Rob,

You wrote:

> *... But I think that an important wider problem that we have to tackle within "reasoning about action and change" is how to synthesise or combine very different approaches to modelling dynamic systems within a single "commonsense" framework. For example, I'd like to see more research along the lines of Erik Sandewall's 1989 work on combining reasoning about actions with modelling using the differential calculus. It's true that there has been a small amount of subsequent work on this theme since then ... But not much compared with, say, work on extending state-transition based approaches to deal with ramifications in evermore sophisticated ways. Why is this so? Why don't we put much effort into addressing challenges such as Kuipers' - on combining the Situation Calculus with Q.R. (see Kuipers' book, p. 201)? ...*

Unfortunately, the answer to this question is a brutal one: publication problems. At least, that's the conclusion I have drawn from the experience of our group. The following is what happened after our start on hybrid systems in 1988-89. A key new result in the 1989 papers [c-kr-89-412], [c-ijcai-89-894] was that minimization or restriction of change generalized nicely to minimization or restriction of discontinuities. (The particular use of chronological minimization as a restrictor on the set of models was of secondary importance, I think; one can do it in other ways). The weak spot that we identified at the same time, and which was clearly spelled out in the papers, was that some additional model selection criterion was necessary, since we still got some unintended models. There were two options: modifying the logic itself, or introducing concepts from other disciplines.

The first approach was pursued by two of our graduate students at the time, Tommy Persson and Lennart Staflin. Their first paper in this direction was accepted at ECAI 1990 [c-ecai-90-497], but then they ran into the wall. One more of their papers is still available as a departmental report; it was called "Cause as an Operation on a Logic with Real-valued Fluents and Continuous Time". The article is available at the URL

`www.ida.liu.se/publications/cgi-bin/tr-fetch.pl?r-90-45+abstr`

and the abstract goes as follows:

> *We propose a new method for characterizing the discontinuities in processes that are mostly continuous. We introduce a causal operator*

that is used to specify when the value of a fluent has a cause. A discontinuity in a fluent is allowed if the fluent's value immediately after the discontinuity has a cause. The causal operator is incorporated in a temporal logic with continuous time and real-valued fluents. The resulting logic is a nonmonotonic logic suitable for representing physical models of real world situations. We define a selection function which given a set of models returns a subset of the models. This selection function defines a nonmonotonic entailment operator. The intuitive idea behind the selection function is that is should select all models where all discontinuities are "specified" as allowed.

In other words, they proposed what is known today as a causal approach. The paper was rejected for IJCAI 1991. Around the same time, my journal style article which combined and extended the results in the 1989 KR and IJCAI papers was rejected for the A I Journal, with vitriolic reviews.

The other approach, which we also investigated, was to bring in aspects of real physics. We started cooperation with people who had that competence, and in particular with our colleagues in control theory. This led to work on the use of bond graphs, which is a classical energy-based method for modelling physical systems, and uniformly applicable to systems from different domains (mechanical, electrical, hydraulic, etc.). Members of our group (Strömberg, Söderman) developed a generalization of bond graphs to take account of abrupt changes (that is, combining continuous and discontinuous) by introducing a "switch" concept in a clean way.

Yet another approach was the use of hybrid transition systems, which are a generalization of the transition systems that come from the theory of real-time systems. Additional members of our group (Nadjm-Tehrani and Strömberg) used hybrid transition systems for modelling actions, analyzing safety conditions ("is it possible that if I drive this way, I may crash into the car in front of me?"), etc.

For both bond graphs and transition systems, the idea was to import methods from other areas into AI and KR. In both cases, our people were able to publish successfully in the neighboring discipline, *but not in AI*, or at least most of the AI submissions were rejected. Reviewers tended to say either that this was not relevant for AI, or that although possibly relevant, more would have to be done in order to reach the presumed high quality standards that we require in our field.

It goes without saying that after a few experiences of this kind, these (then) Ph.D. students turned away from AI and continued their work in the areas where they were better received. They were also put off by what they considered as idiotic comments by reviewers, for example, to the effect that the proposed modelling system was not capable of accounting for the sudden occurrence of asteroids on the scene.

When these things happen, it is our discipline that stands to lose. There were great opportunities at that time for bringing in fresh concepts into KR, and for integrating them with what we are otherwise doing. On the other hand, time does not stand still while we fumble, and if our area does not deal in a timely fashion with new problems, then there are others who will.

It is also important to note that this resistance to new ideas is not reciprocal. This year's HART conference (Hybrid And Real Time systems) had no trouble accepting my paper on relating high-level and low-level descriptions of actions, which was an extension of my invited paper at last year's ECAI.

This panel discussion has already touched on the remarkable persistence of situation calculus in our field. The field's unwillingness to accept and use outside knowledge for dealing with continuous change is equally remarkable.

Erik

References:

[c-ecai-90-497] Tommy Persson and Lennart Staflin. *A Causation Theory for a Logic of Continuous Change.* Proc. European Conference on Artificial Intelligence, 1990, pp. 497-502.

[c-ijcai-89-894] Erik Sandewall. *Filter Preferential Entailment for the Logic of Action in Almost Continuous Worlds.* Proc. International Joint Conference on Artificial Intelligence, 1989, pp. 894-899.

[c-kr-89-412] Erik Sandewall. *Combining Logic and Differential Equations for Describing Real-World Systems.* Proc. International Conf on Knowledge Representation and Reasoning, 1989, pp. 412-420.

From: Patrick Doherty on 17.11.1997

After following the discussion between Rob Miller and Tom Costello, I'd like to point out another approach that my group has been using in our research in the area of action and change. It is based on a distinction between surface and base languages made by Sandewall in Features and Fluents. The family of logics we use is called TAL (Temporal Action Logics) and the newer versions are generalizations of an entailment policy called PMON, first described in F&F.

Clearly, high-level narrative description languages are not only useful, but will obviously be necessary when dealing with scenarios more complex than those we see in the literature today. On the other hand, general purpose logics such as classical logic have great advantage when doing comparative analyses, debugging and incremental extension of formalisms.

In our approach, we combine the advantages of each. Our narrative descriptions are represented in terms of a high-level language which allows for straightforward description of observations, action instances and types, casual rules, and explicit temporal constraints. The high-level language may simply be viewed as a set of macros where each has a modular translation into formulas in the base language, 1st-order classical logic. The language is always extensible in an incremental manner. So far, we've extended the language for causal rules and concurrency simply by adding new macros and translation functions.

The logic TAL 1.0, and the approach using surface and base languages is implemented and accessible as an applet or a Marimba Castanet Channel. The visualization tool allows for the construction of narratives in the high-level language, their automatic translation into a 2nd-order theory, and that theories automatic translation into a first-order theory. One also has the possibility of viewing models as time-lines and a query mechanism is provided. The system and related references are accessible via the following URL:

`http://anton.ida.liu.se/vital/vital.html`

The majority of scenarios discussed in the literature are represented in the tool and can be queried. The purpose of the tool is not only for our individual research, but also to open up the logics for public evaluation and comparative analyses. The use of both a high-level macro language and a translation into classical logic should meet the needs of groups taking the $\mathcal{A}$ language approach or those more comfortable with good old classical logic.

The danger we find with the trend in using $\mathcal{A}$ language approaches is that it often appears to be the case that one is taking a relatively simple surface language and translating into what turns out to be something along the lines of classical logic, but in a rather indirect and complex manner. It is difficult to see how the guarantee of semantic continuity in the base language or incrementality in the surface language will be met as scenarios or narratives become increasingly more complex. On the other hand, if provided with well-understood and modular translations into classical logic, it is much easier to evaluate progress and simplify comparisons. One sign that there is a problem is that the $\mathcal{A}$-type languages are generally only compared relative to other $\mathcal{A}$-type languages. Of course, translations of formalisms to classical logic and ensuing comparisons are not all that simple when comparing widely differing ontologies, but we have a rich infrastructure of well-established technical tools to help us along.

I'm certainly all for the flourishing of alternative approaches to modeling action and change, but I really think it is time to clean up our methodology, do more comparative analyses across paradigms regarding strengths, weaknesses, assessments of use, and to apply the formalisms to some "real" problems in the area of DES, process control, etc. I'd like to see a library of tools and implementations which allow each of the different groups to actually test the representational capabilities of the perspective approaches and a number of realistic modeling challenges similar to those one finds in the "Hybrid Systems" area. This appears to be a common and useful aspect of methodology in other areas. Why is this lacking in our area and what can we do about it?

I hope the tool we have developed and placed on-line might serve as a starting point for discussion or for developing healthier methodological tools and coherence in the area. Perhaps Murray Shannahan's and Ray Reiter's interests in controlling robots with logics could also lead to another set of testbed's for comparative analysis of formalisms.

From: Vladimir Lifschitz on 19.11.1997

I would like to respond to some of the comments on action languages published in ENRAC 13.11 and 17.11.

Tom Costello writes to Tony Kakas and Rob Miller regarding their new action language:

> *The reason I ask for truth conditions for your propositions is that I cannot understand what the intuitive consequences of a set of propositions should be, unless I understand what the propositions say.*
>
> *As you say, action languages are supposed to be "understandable and intuitive". Languages cannot be understood without semantics.*

It seems to me that the semantics of an action language cannot be described by specifying truth conditions for its propositions. The problem is the same as with nonmonotonic languages in general. Take, for instance, the closed-world database P(1),P(2). The negation of P(3) is a consequence of this database, but this fact cannot be justified on the basis of truth conditions for P(1) and P(2).

Patrick Doherty writes:

> *The danger we find with the trend in using $\mathcal{A}$ language approaches is that it often appears to be the case that one is taking a relatively simple surface language and translating into what turns out to be something along the lines of classical logic, but in a rather indirect and complex manner.*
>
> *On the other hand, if provided with well-understood and modular translations into classical logic, it is much easier to evaluate progress and simplify comparisons.*

It is impossible, unfortunately, to translate an action language into classical logic in a modular way, because classical logic is monotonic, and action languages are not. The best we can achieve is a modular translation from an action language into a nonmonotonic formalism, such as circumscription, whose semantics can be characterized by a nonmodular translation into classical logic. This is indeed indirect and complex. But we have to pay this price for the convenience of reasoning about actions in classical logic.

Vladimir Lifschitz

From: Erik Sandewall on 21.11.1997

Vladimir,

In ENRAC 19.11, in the context of the discussion with Tom Costello and Patrick Doherty, you wrote:

> *It is impossible, unfortunately, to translate an action language into classical logic in a modular way, because classical logic is monotonic, and action languages are not. The best we can achieve is a modular translation from an action language into a nonmonotonic formalism, such as circumscription, whose semantics can be characterized by a nonmodular translation into classical logic. This is indeed indirect and complex. But we have to pay this price for the convenience of reasoning about actions in classical logic.*

Through the Features and Fluents approach, we have demonstrated a much more direct and less complex way of doing these things. However, "convenience of reasoning" is not the only issue, and it's not what Patrick addressed; what he actually wrote and what you quoted in the preceding lines was:

> *On the other hand, if provided with well-understood and modular translations into classical logic, it is much easier to evaluate progress and simplify comparisons.*

In particular, translating scenario descriptions into first-order logic helps evaluation and comparisons in two ways. First, for transparency, i.e. for allowing us to understand in a precise manner what the scenario descriptions say, which is also what Tom Costello's persistent questions are concerned about. After it, there is the issue of actually carrying out the reasoning about actions, which quite possibly can be done (or implemented) more conveniently if one uses first-order theorem provers as inference engines. Anyway, let's stick to the question of how we can evaluate progress and simplify comparisons for the work that gets done and published.

With respect to transparency, it seems to me that action languages such as $\mathcal{A}$ and $\mathcal{E}$ add to the obscurity rather than dissolving it. In the Features and Fluents approach, we achieve the same results in a much less elaborate fashion: we have *one single language*, namely a straightforward multi-sorted first-order theory; we have a set of *syntactic abbreviations* or "macros" in order to sugar the notation for legibility; and we have *two different semantics*. There is the classical semantics of the Tarskian type which is almost straight from the textbook, with routine modifications to take care of the multitude of types and for assuring a closed-world assumption with respect to objects. There is also the *underlying semantics* which specifies what one really means - corresponding to Tom Costello's question to Tony Kakas and Rob Miller, where he wrote:

> *The reason I ask for truth conditions for your propositions is that I cannot understand what the intuitive consequences of a set of propositions should be, unless I understand what the propositions say.*
>
> *As you say, action languages are supposed to be "understandable and intuitive". Languages cannot be understood without semantics.*

The underlying semantics does exactly this. To put it another way, here is a recipe for converting an action-language approach to our approach. You take the action-language setup with its two languages, each with its own syntax and semantics. First, you remove the separate syntax of the action language. You retain the use of two distinct semantics, so one and the same scenario description can be mapped into a set of models by two different methods: the set of classical models, and the set of intended models. Also, you make sure that the two semantics use the same space of possible interpretations; it's just that the set of intended models is (usually) a subset of the set of classical models. The you have captured the essentials of our approach.

The definition of the underlying semantics is indeed not truth-functional in a conventional way; it can rather be described as a kind of simulation of the world in question. However, truth conditions are still used within that 'simulation', namely for defining the truth conditions for each individual observation statement. The resulting semantics is concise, intuitively convincing, and formally precise at the same time.

This approach has several important advantages:

- No need to define new languages all the time, and of comparing newly published languages with previously published ones. We stay with the same language, which is sufficiently expressive right from the start to last for a while. In particular, it uses an explicit time domain and allows both non-metric "successor" time, integer time, and real time. The time domain may be either linear or forward-branching. Multi-valued fluents are allowed; objects are allowed so the language is "first-order" rather than "propositional", and others more.
- New problems can be addressed with very small initial effort.

Consider ramification, for example. In the $\mathcal{A}$ tradition, new language variants were introduced for ramification. In our approach, all you have to do is to remove the syntactic restriction that is imposed in the footnote on page 176 in the "Features and Fluents" book, select PMON among the twelve entailment methods that are defined and analysed there, and you have a method for ramification that in fact is as powerful or more powerful than most of what has been published in the last couple of years. (To be precise, the two formulations of PMON that are stated on page 243 are no longer equivalent after the generalization, and you have to use the second one. That's all).

- Consequently, you can get much more quickly to the point where you actually *analyse* the entailment methods in order to verify their range of applicability. You don't spend so much of your time setting up the definitions.

A possible objection against defining a quite expressive language from the start is that you may not be able to prove your results for such a broad language. That is one reason why we focus on "range of applicability" results rather than "validation" results. Instead of defining a narrow language and proving that a property holds throughout the language, we identify what is the part of our broad language where the property holds. This helps avoiding the proliferation of languages, but it also helps obtaining results that are as general as possible, since the result is not artificially constrained by the choice of language. This is a real difference between the approaches, since the published general results about $\mathcal{A}$ type languages are consistently validation results (unless I have missed something).

Then there is the other reason for reducing an action language to a first-order theory, namely for implementation purposes. There, again, Doherty et al have developed efficient computational mechanisms as a part of our approach; their implementation has been available for on-line use over the net since May of this year. In their case it's only an implementation issue; the purpose of their work is not to assign a meaning to the language since that has already been taken care of. (Patrick writes more about this in his contribution to the discussion, below).

The bottom line is that it is perfectly possible to achieve legibility (initial motivation of action languages), transparency (Tom Costello's request), and effective implementation by the approach used in Features and Fluents, and with much less administrative overhead than in the work based on action languages. My question is, therefore: what are the real benefits of all the definitional work in the action-language papers; what are the results that the rest of us can use?

From: Tom Costello on 21.11.1997

Vladimir writes,

> *It seems to me that the semantics of an action language cannot be described by specifying truth conditions for its propositions. The problem is the same as with nonmonotonic languages in general. Take, for instance, the closed-world database P(1),P(2). The negation of P(3) is a consequence of this database, but this fact cannot be justified on the basis of truth conditions for P(1) and P(2).*

I do not ask that Action language designer's define their semantics in terms of truth conditions. I ask that the designers give truth conditions to each of their assertions. The difference can be seen in your example if you take P to mean there is a flight and 1 to means Glasgow,London and 2 to mean London,Moscow. The P(1) is true, if there is a flight from Glasgow to London. What is puzzling me about Kakas and Miller's language is what their propositions mean, in exactly this sense.

In particular, what does

```
A causes F if G
```

or

```
A initiates F if G
```

mean. That is, given a model M of a domain description, when is this proposition satisfied in M. I have pointed out that problems arise under certain definitions of model.

The most difficult issue that I am aware of, is that it is unclear whether

```
A causes F if G
```

means that

```
in every actual state S where G is true,
    then F is true in R(A,s),
```

or the similar, but different

```
in every possible state S where G is true,
    then F is true in R(A,s).
```

Similarly, does

```
Always F,G
```

or Kakas and Miller's

```
F Whenever -G
```

mean that every actual state satisfies F,G, or every possible state.

Tom Costello

From: Patrick Doherty on 21.11.1997

Vladimir's reply to my paragraph about $\mathcal{A}$ Languages introduces two interesting and subtle issues:

1. What is meant by modular/non-modular translation.
2. What is meant by a monotonic/nonmonotonic approach.

I am not sure if these topics belong to the ontology panel, but they are highly relevant when dealing with comparative analyses across formalisms and understanding what on the surface appear to be opposing "ideological stances" in methodology.

To clarify my "current" working view of the role action languages should play, I simply quote the original intuition Vladimir himself at one time had about their role in his paper, "Two Components of an Action Language":

> *Originally, action languages were meant to play an auxiliary role. The primary goal was to represent properties of actions in less specialized formalisms, such as first-order logic and its nonmonotonic extensions, and the idea was to present methods for doing that as translations from action languages.*

My group currently uses this approach and it is also one of the cornerstones of the Features and Fluents methodology. Formally and conceptually, we translate from an action scenario description language into a first-order theory with a circumscription axiom. I consider 2nd-order theories to be part of classical logic. From an analysis of the circumscriptive theory, we can identify different means of deriving efficient computational mechanisms for reasoning or "querying" a class of action scenarios.

The advantages we have derived from this approach are the following:

1. A separation of the ontological and epistemological analysis from the generation of computational procedures for querying action scenarios.
2. A direct means of comparing the ontological and epistemological assumptions we make with those of others, including across paradigms.
3. A somewhat less-direct means of employing part or all of particular computational mechanisms proposed by other groups, such as the use of "extended logic programs", regression techniques, or "explanation closure" approaches, for example.

This brings me to the more specific topics of modularity in translation and monotonicity/nonmonotonicity.

As regards modularity:

In our current family of logics (TAL), we find that the circumscription policies used are really quite simple, basically nothing more than predicate completion. This given, we can either reduce the 2nd-order theory associated with an action scenario using a reduction algorithm in a "nonmodular" manner, or generate a "modular" translation directly from the action scenario description which includes one additional formula for each predicate completion. So, I'd disagree with Vladimir's view on the coupling between modular/nonmodular – monotonic/nonmonotonic in his reply. Although one can interpret the use of a reduction algorithm as nonmodular, one can also interpret the use of local syntactic translation as modular. Again, it all comes down to what is meant by "modular" and we may have slight disagreements on pinning down a definition.

As regards the monotonicity/nonmonotonicity issue:

One fascinating insight which the translation into a circumscribed theory provides us with, is that the automated reduction of the circumscription axiom to a logically equivalent 1st-order formula, essentially generates what could be interpreted as "explanation closure axioms". This has been noted and used by other researchers in other contexts such as Vladimir, Ray Reiter, and Fangzen Lin, although in these cases, one works directly with syntactic translations on an existing 1st-order theory rather than direct translation from a circumscription formula.

So this could turn out to be a non-issue in the sense that meta-assumptions of a nonmonotonic character are sometimes left outside a formalism, but guide the way we write axioms or use syntactic translations, or the assumptions are part of the actual formal theory as in the case of using circumscription axioms or default inference rules. There should be a straightforward means of providing formal translation between the two "stances". The discusion would than revolve around what criteria, such as elaboration tolerence or efficient theorem-proving methods, contribute to a particular choice of "stance".

One other related point worth discussion is that if one takes a good look at the diverse approaches being proposed and actually applied in some sense of the word "applied", analysis at the classical logic level shows that

1. the minimization policies are very similar to each other, even across ontological choices. Currently, we are not doing much more than a somewhat "enhanced" form of predicate completion.
2. The language fragment used is generally not more than an "enhanced" Horn fragment.

Not surprising, because we want to develop efficient query mechanisms, be they logic programs or non-standard procedural mechanisms.

This is why I like approaches which relate in a direct manner to classical logic. We can say useful things like:

- "the 'occlude', 'release', and 'noninert' predicates relax an overly strong minimal change policy by importing the 'varied' part of a circumscription policy into the object language. This results in computational benefits due to the resulting simplified circumscription policy actually used. The approach also provides a straighforward technique for modeling non-determinism."
- "filtered preferential entailment and nested circumscription are useful technical tools for increasing the granularity at which minimization can be applied to action theories, but with the added danger of introducing non-consistency preserving formalisms when distinguishing between observations and action occurrences."

These techniques have informal correlates in the original work by McCarthy and have been refined into formal techniques used by a number of researchers in this area. The problem is that this type of analysis is rare. A contributing factor to the lack of generic analyses could very well be the diversity of specialized action languages and the often complex and direct translations to procedural or computationally oriented frameworks.

From: Pat Hayes on 25.11.1997

Vladimir Lifschitz writes:

> *It seems to me that the semantics of an action language cannot be described by specifying truth conditions for its propositions. The problem is the same as with nonmonotonic languages in general. Take, for instance, the closed-world database P(1),P(2). The negation of P(3) is a consequence of this database, but this fact cannot be justified on the basis of truth conditions for P(1) and P(2).*

But it can be justified on the basis of the truth conditions for P(3), which is just as much a proposition of the language as the first two. Nonmonotonic logics are not classically truthfunctional, but they do have truth conditions. The simple language sketched here has the truth condition: P(n) is true iff P(n) occurs in the database.

Pat Hayes

From: Vladimir Lifschitz on 27.11.1997

Erik,

In ENRAC 21.11 you discuss advantages of the F&F approach, in comparison with action languages, and you write:

> *No need to define new languages all the time, and of comparing newly published languages with previously published ones. We stay with the same language, which is sufficiently expressive right from the start to last for a while ...*

I'd like to understand this better. The need to define new action languages arises when we want to describe aspects of reasoning about action that have not been understood in the past. Here are some examples.

1. Ramification constraints vs. qualification constraints. A fact about fluents sometimes allows us to conclude that an action has an indirect effect, and sometimes that it has an implicit precondition. (It functions sometimes as a "ramification constraint" and sometimes as a "qualification constraint.") Example: the objects that I have in my pocket are in the same place where I am. After I come home with a comb in my pocket, the comb will be in my home also; this is an indirect effect. Since knives are not allowed in airplanes, I can't board an airplane with a knife in my pocket; this is an implicit precondition.

2. Asymmetry of ternary constraints. Consider a spring-loaded suitcase with two locks. Its state can be described by three fluents: lock 1 is open; lock2 is open; the suitcase is closed. The constraint is that these fluents cannot hold simultaneously. Consider a state in which one of the locks is open and the suitcase is closed. When I open the other lock, this causes the suitcase to open. (This action does not cause the first lock to close, which, logically speaking, is another possibility.)

3. Interaction between concurrently executed acions. Consider lifting the opposite ends of a table upon which various objects have been placed. If one end of the table has been raised, the objects on the table will fall off. But if both ends are lifted simultaneously, the objects on the table will remain fixed.

These phenomena could not be described in the original action language A and in some of its successors. New, more expressive languages had to be designed. I am wondering what the status of examples 1-3 in the F&F framework is. Would you be able to formalize them in your original language, which you described as sufficiently expressive right from the start? My understanding of the possibilities of F&F is not sufficient to answer this without your help. But about the situation calculus I know that new syntactic features had to be added to it to address these problems, such as the predicate *Caused* (similar to *Holds*, but not quite the same), and an addition operation on actions (to represent concurrent execution).

Patrick,

In ENRAC 21.11 you write:

> *To clarify my "current" working view of the role action languages should play, I simply quote the original intuition Vladimir himself at one time had about their role in his paper, "Two Components of an Action Language":*

"Originally, action languages were meant to play an auxiliary role. The primary goal was to represent properties of actions in less specialized formalisms, such as first-order logic and its nonmonotonic extensions, and the idea was to present methods for doing that as translations from action languages."

My group currently uses this approach and it is also one of the cornerstones of the Features and Fluents methodology.

Indeed, your TAL is essentially an action language. There are minor differences in style between TAL and the languages that I've been working on. What you write as

```
[t1,t2] move(p,l) ~>
  [t1] !(place_of(p) == l) -> [t1,t2] place_of(p) := l
```

I would maybe represent this as

```
move(p,l) CAUSES place_of(p)=l,
IMPOSSIBLE move(p,l) IF place_of(p)=l.
```

This is slightly more concise because t1, t2 are suppressed. I am wondering whether you would lose any important expressivity if you changed your macros in a similar way.

Tom,

In ENRAC 21.11 you write:

The most difficult issue that I am aware of, is that it is unclear whether

```
A causes F if G
```

means that

```
in every actual state S where G is true,
      then F is true in R(A,s),
```

or the similar, but different

```
in every possible state S where G is true,
      then F is true in R(A,s).
```

I would say it's the latter.

Pat,

In ENRAC 25.11 you write:

Vladimir Lifschitz writes:

> It seems to me that the semantics of an action language cannot be described by specifying truth conditions for its propositions. The problem is the same as with nonmonotonic languages in general. Take, for instance, the closed-world database P(1),P(2). The negation of P(3) is a consequence of this database, but this fact cannot be justified on the basis of truth conditions for P(1) and P(2).

> *But it can be justified on the basis of the truth conditions for P(3), which is just as much a proposition of the language as the first two.*

Good point. Our closed-world database determines the model in which P(3) is to be evaluated according to the truth conditions of classical logic.

Similarly, a domain description in the language $\mathcal{A}$ determines the transition diagram in which a value proposition is to be evaluated when we want to determine whether it is a consequence of the description.

Further you write:

> *Nonmonotonic logics are not classically truthfunctional, but they do have truth conditions.*

Here I cannot fully agree with you. A default theory in the sense of Reiter is defined by its axioms and its defaults; we have truth conditions for axioms, but not for defaults.

Vladimir