Reasoning About Actions in Narrative Understanding

Srinivas Narayanan
CS Division, UC Berkeley and ICSI
snarayan@jcsJrsi}.herkeley.edu

Abstract

Reasoning about actions has been a focus of interest in AI from the beginning and continues to receive attention. Rut the range of situations considered has been rather narrow and falls well short of what is needed for understanding natural language. Language understanding requires sophisticated reasoning about actions and events and the world's languages employ a variety of grammatical and lexical devices to *construe, direct attention and focus* on, and *control inferences about* actions and events. We implemented a neurally inspired computational model that is able to reason about, linguistic action and event descriptions, such as those found in news stories. The system uses an active. event representation that also seems to provide natural and cognitively motivated solutions to classical problems in logical theories of reasoning about actions. For logical approaches to reasoning about actions, we suggest, that looking at story understanding sets up fairly strong desiderata both in terms of the fine-grained event and action distinctions and the kinds of real-time inferences required.

1 Introduction

Formal approaches to model reasoning about changing environments have a long tradition in AI. This research area was initiated by McCarthy [McCarthy, 19(H)], who claimed that reasoning about actions plays a fundamental role in common sense. Trying to build language understanding programs not only underscores the importance of reasoning about actions, but also suggests that the the set of situations and the kinds of inferential processes required are richer than has been traditionally studied in formal approaches.

Language understanding requires sophisticated reasoning about actions and events. The world's languages have a variety of grammatical and lexical devices to *construe, direct attention and focus,* and *control inferences* about, actions and events. Consider the meaning of *stumbling* in the following newspaper headline "Indian Government stumbling in implementing Liberalization Plan'' Clearly, the speaker intends to specify that the liberalization plan is experiencing some difficulty. Moreover, the grammatical form *is + VP-ing* suggests that the difficulty facing the plan is ongoing and the final outcome of the plan is indeterminate. Compare this to the subtle meaning differences with grammatical and lexical modifiers on the same root verb such as *has stumbled* or *starting to stumble.* Most readers are likely to infer after reading this sentence that the government's liberalization policy is likely to fail, but this is only a default causal inference that is made in the absence of information to the contrary. Finally, how does *stumble,* whose basic meaning is related to spatial motion and obstacles get interpreted in a narrative about international economic policies?

We have implemented a computational model that is able to reason about action and event descriptions from discourse fragments such as the one above. The system uses an active event representation that also seems to provide natural and cognitive!}- motivated solutions to classical problems in logical theories of reasoning about, actions. We first present the main features of our representation and show that if provides a computational model for existing formalisms for reasoning about actions. We then suggest how looking at story understanding sets up fairly strong desiderata for logical approaches to reasoning about actions both in terms of the fine-grained event and action distinctions and the kinds of real-time inferences required.

2 The Action Model

Our action theory comprises of two central components; 1) an executing representation of *actions* (called x-schemas) based on extensions to Petri Nets and 2) a Belief Net model of state that captures and reasons about, complex dependencies between state variables.

2.1 An Executing Semantics of *Actions*

We represent actions as a modified class of high-level Petri Nets [Reisig, 1985] called x-schemas. The most relevant features of Petri nets for our purposes are their ability to model events and states in a distributed system and the clean manner in which they capture sequentiality, concurrency and event-based asynchronous control.

Definition 1 . The basic x-schema: An x-schema consists of *places(P)* and *Transitions (7)* connected by weighted directed arcs $\mathcal{A}$ ($\mathcal{A} \in (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$) Each arc $a_{ij} \in \mathcal{A}$ has weight w_{ij} ($w_{ij} \in \mathcal{N}$) Input Arcs $*\mathcal{T}$ ($*\mathcal{T} \in (\mathcal{P} \times \mathcal{T})$) connect Input Places to Transitions. Output Arcs 7* ($\mathcal{T}* \in (\mathcal{T} \times \mathcal{P})$) connect Transitions to Output Places. Arcs are typed as *enable* arcs $\mathcal{E}$, *inhibitory* arcs I, or *resource* arcs 7v .

X-schemas have a well specified real-time execution semantics where the next state function is specified by the firing rule. In order to simulate the dynamic behavior of a system, a Marking (distribution of tokens in places (depicted as dark circles or numbers)) of the x-schema is changed according to the following firing rule.

Definition 2 . Execution Semantics of the basic x-scheina A transition T is said to be enabled if no *inhibitory* arc $i \in \mathcal{I}$ of $\mathcal{T}$ lias a marked source place and all sources of enable arcs $e \in \mathcal{E}$ of $\mathcal{T}$ are marked and all input arcs $p \in \mathcal{R}$ have at, least w_{pt} tokens at their source place, where w_{pt}. is the weight of the arc from $\mathcal{P}$ to $\mathcal{T}$ The firing of an enabled transition T , removes *wpf* tokens from the source of each non-inhibitory, non-enabled input arc *V* and places wrp tokens in each output place of T .

X-schemas cleanly capture sequentially, concurrency and event-based asynchronous control; with our extensions they also model *hierarchy, stochasticity* and */>a-ranietenzation* (run-time bindings). Besides *typed arcs* (Definition 1), the following two extensions to the basic Petri net are designed to allow us to model hierarchical action sets with variables and parameters:

First., tokens carry information (i.e. they are individuated and typed) and transitions are augmented with predicates which select tokens from input places based on the token type, as well as relate the. type of the tokens produced by the tiring to the types of tokens removed from the input.

Second, transitions are typed. Figure 1 shows the four types of x-schema transitions, namely stochastic, durative, instantaneous and hierarchical transitions. An instantaneous transifion(shown as dark rectangles) *fins* as soon as it is enabled. A timed transition (shown as rectangles) fires after a fixed delay or at an exponentially distributed rate. Hierarchical transitions (depicted as hexagons), activate a subnet., wait for its return, or time-out.

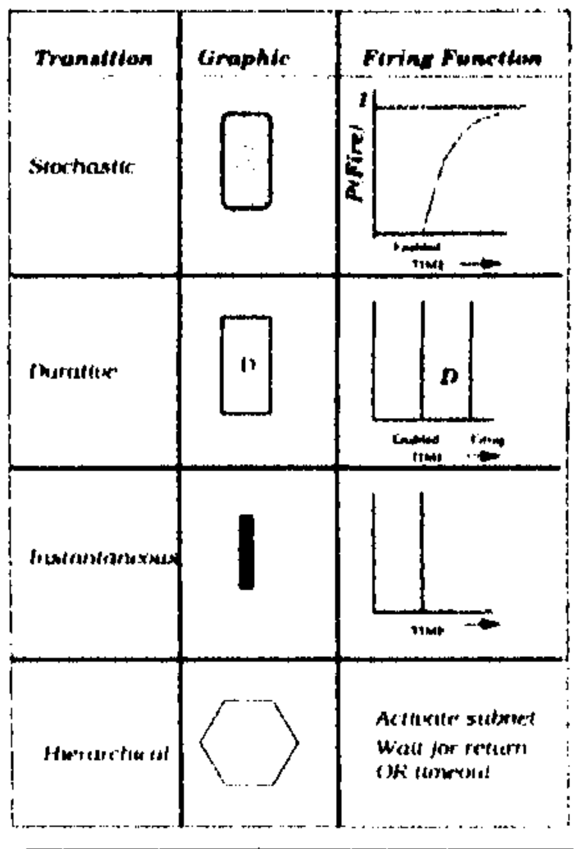

Figure 1: Basic types of transitions.

Theorem 1 . *[Narayanan, 1997] An j-schema is formally equivalent to bounded High Level Generalized Stochastic Petri Net (HLGSPN). The reachability graph of a marked x-schema is isomorphic to a semi-markwv process.*

2.2 A Belief Net Model of States

Our representation of states must, be capable of modeling causal knowledge and be able to support both belief updates and revisions in computing the global impact of new observations and evidence both from direct observations and from action effects. Our implementation of the agent's state uses Belief Networks [Jensen, 1996J.

Belief networks consist of a *net* of variables and a set, of directed links. Each variable lias a finite set of mutually exclusive states. The variables together form a *DAG* (Directed Acyclic Oraph). To each variable *A* with parents *B\ . . . B_n* there[1] is attached a conditional probability table $P(A|B_1, \ldots B_n)$.[1]

For a Belief Net B_u , the following theorem allows us to calculate the joint probability *P(V)* from the conditional probabilities in the network.

Theorem 2 . The Chain Rule [J*ensen, 1996] Let B_u be a BN over $U = (A_1 \ldots A_n)$l . Then the joint probability distribution P(U) is the product of all conditional probabilities specified in B_u*

$$P(U) = \prod_i P(A_i|pa(A_i)) \qquad (1)$$

where $pa(A_i)$ is the parent set of A_i

2.3 Temporal Projection

We now turn to one of the central issues in reasoning about action, i.e. the temporal projection problem. The

[1] Note that the *GFT* entries could instead use the ranking function method of K -calculus [Golds/midt, 1992], where probabilities are mapped onto a quantized logarithmic scale which forms the basis of a ranking function.

problem consists of computing a final state s_{n+1} that results from executing the action set $\mathcal{S} = [a_1, \dots a_n]$ in a given initial state s_0 • The solution to this problem in our action model follows.

Algorithm 1 Temporal Projection

1. Set initial Marking M_0. $\forall p \in s_0 : M_0(p) = 1, \forall p \notin s_0 : M_0(p) = 0$
2. Fire enabled transitions $T_{e_0} \in T$ of M with initial marking Mo • The next state function described earlier takes the system to a new marking M_{into} ■ The state corresponding to this marking $S_{int} = \forall p : M_{int}(p) = 1 : p \in S_{int}, \forall p : M_1(p) = 0 : p \notin s_{int}$
3. Run the belief propagate procedure to return the most consistent aposteriori assignment (*MAP)* of values to the state variables. The new state s corresponds to the marking M_1 where $\forall p \in S_1 : M_1(p) = 1, \forall p \notin s_1 : M_1(p) = 0$.
4. **While** $1 \leq i \leq n$, **do**: Fire enabled transitions $T_{e_{i-1}}$ with marking M_i. set $M_{int} = *a_i \bigcup M_i$. Run Step 3 to get S_{i+1}. **Return** S_{n+1} as the answer.

Steps 1, and 2 are essentially constant time, since our notion of state as a graph marking is inherently distributed over the network, so the working memory of an x-schema-based inference system is distributed over the entire set of x-schemas and state features. The result is a massively parallel solution to the projection problem. Step 3 requires belief propagation which is well known to be intractable for complex domain topologies. So in our model, executing actions is fast., parallel and reflexive, while propagating indirect effects with complex domain dependencies to achieve global consistency is hard. In addition, the central features of our action representation, namely that they are *executing* provides an elegant solution to the Frame Problem. Specifically, the action-based executing action semantics allows frame axioms to be implicitly encoded in the structure of the net and the local transition firing rules.

3 Modeling Action Theories

Although the mechanisms outlined above were developed for language understanding, they seem to be useful for some of the problems discussed in the recent literature. We assume the reader is familiar with reasoning with the syntax of action models similar to [Gelfond & Lifschitz, 1993]. To keep the exposition simple, we will consider only propositional fluents and deterministic actions. As in *ARV* [Giunchiglia & Lifschitz, 1995], we model both "inertial" (always C (where C is a formula)) and "dependent" (A dependsJOII B) fluents. The following rules present the basic encoding of action theories (assuming the syntax of the *ARV* theory) in our model.

1. Static Fluent names are places. Actions names are Transition labels. Preconditions are pre-sets (*T), direct effects are post-sets (T*) of transitions. If the truth-value of a fluent $f \in \mathcal{F}$ is *true* , in State S_i , then the marking M,-(/) = 1 ?
2. Domain Constraints with inertial fluents are modelled as instantaneous transitions. Statements in *ARV* , of the form always $A \supset B$, add an instantaneous transition with pre-set $*\mathcal{T}_{AB} = A$, post-set $\mathcal{T}_{AB}* = A, B$. Dependent fluents are modeled as arcs in the Agent state belief net. More precisely, the statement f_j depends on f_i if f_k , results in an arc from the variables representing f_i, f_k to f_j The *CPT* entries for f_j are given by the appropriate constraints (including prior knowledge).
3. Initially C, is modeled by assigning an initial marking where $\forall f \in \mathcal{F} \bigcap C : M_0(f) = 1$. $\forall f_i, fj \in F$, if f_i dependsjon *fj* , add an instantaneous transition $\mathcal{T}_{\mathcal{I}\mathcal{J}}$ with preset $*\mathcal{T}_{\mathcal{I}\mathcal{J}} = f_j$ and post set $\mathcal{T}_{\mathcal{I}\mathcal{J}}* = f_i, f_j$

We now look at two standard examples from the literature that illustrate some issues with reasoning about actions. One is the standard "potato in the tailpipe" problem, and the other is the "jumping into lakes" problem[Giunchiglia & Lifschitz, 1995].

Example 1 Potato in Tailpipe The state fluents here are *potato* (potato in the tailpipe), *clog* (tailpipe is clogged). The actions are *Put P), Remove _P)* (put/remove potato in/from tailpipe), *Start* (start the engine). The domain is characterized as *Put_p)* causes *potato, Remove 4)* causes *-^potato Start* causes *running* , *clog* dependsjon *potato,* always *potato* $\supset$ clog. ■

Example 2 Wetness Here the idea is that Jumping into a lake (*JurnpTn*) has the direct effect of being *InLake* , and the indirect of making you *Wet* . Jumping out gets you out of the lake, but you are still wet if you were in the lake. The domain is described in *ARV* as always Inlake $\supset$ *Wet* , *Jump In* causes *InLake* , *JumpOut* causes *-InLake* , Initially *-^InLake.* . ■

The reader can easily verify that the construction rules above result in the models shown in Figure 3 and Figure 2 for Example 2 and Example 1 respectively.

Proposition 1 . *The procedure above results in a* causal model *for a domain description D (in the Syntax of ARV) in that it satisfies all the causal laws in D . Furthermore, a value proposition of the form C* after *A is entailed by D* **iff** $\forall c \in C, c \in S_i$ *where* S_i *is the state that results after running the projection algorithm on the action set A .*

[2] In general, the representation allows states with types and integer measure fluents, in which case the multi-set representing the place would be marked by the appropriate number and type of tokens.

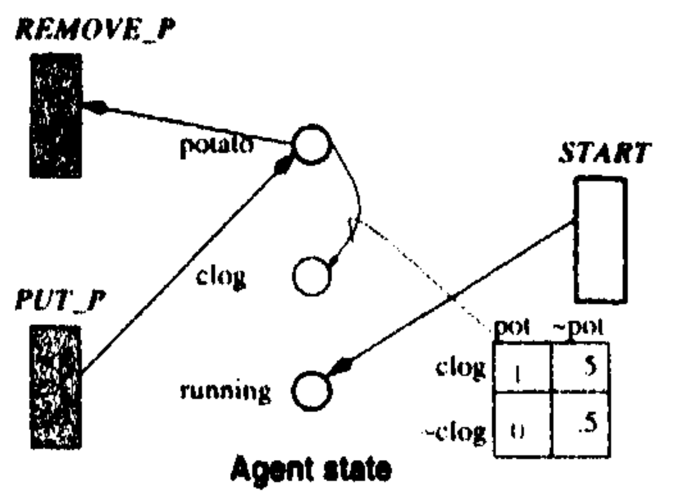

Figure 2: Potatoes in Tailpipes

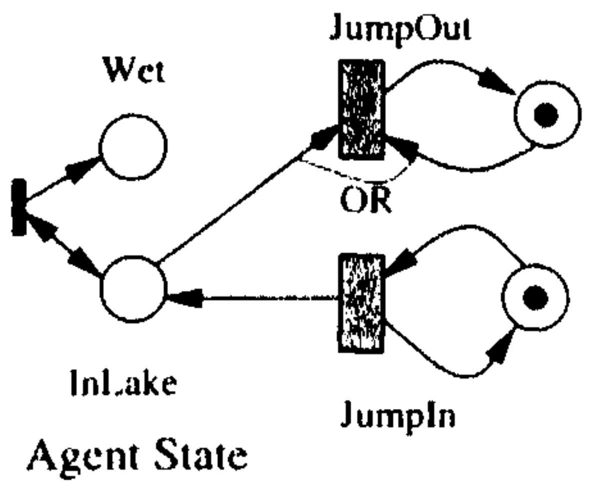

Figure 3: Jumping Into Lakes

3.1 Ramifications, Inertial and Dependent Fluents

Indirect effects are quite naturally handled by our system. Indirect effects that are "inertial fluents" get set by instantaneous transitions. In Figure 3, the direct effect, of Jumpln sets the fluent InLake, which instantaneously sets the value of the fluent Wet. Note that the fluent Wet persists unless some other action is taken (like drying) to change the value of this fluent. In contrast, note that in Figure *2,* the truth value of the 'dependent fluent" clog is determined by the fluent potato. While the value proposition $\neg potato \wedge \neg clog$ after *put-pat* ; *remove-pot* , is not entailed by the description depicted in Figure 2 (from the uniform prior for $P(clog|\neg potato)$); if the domain theory contains the explicit knowledge initially $\neg clog \wedge \neg potato$, the proposition is entailed by the model.[3]

4 Understanding Language About Actions and Events

The frequency with which languages refer to events, the universality of such expressions, and the subtlety in the kinds of distinctions made have made the temporal character of events in language (called linguistic aspect) an object of study since Aristotle. Somewhat more recently, the complex and context-sensitive determination of aspectual status, or the internal temporal shape of an event has been the focus of much work [MS, 1988].

Many languages have a variety of grammatical aspectual modifiers such as the English *progressive* construction

[3]More generally, logical entailment can he viewed as the downward closure of the final marking.

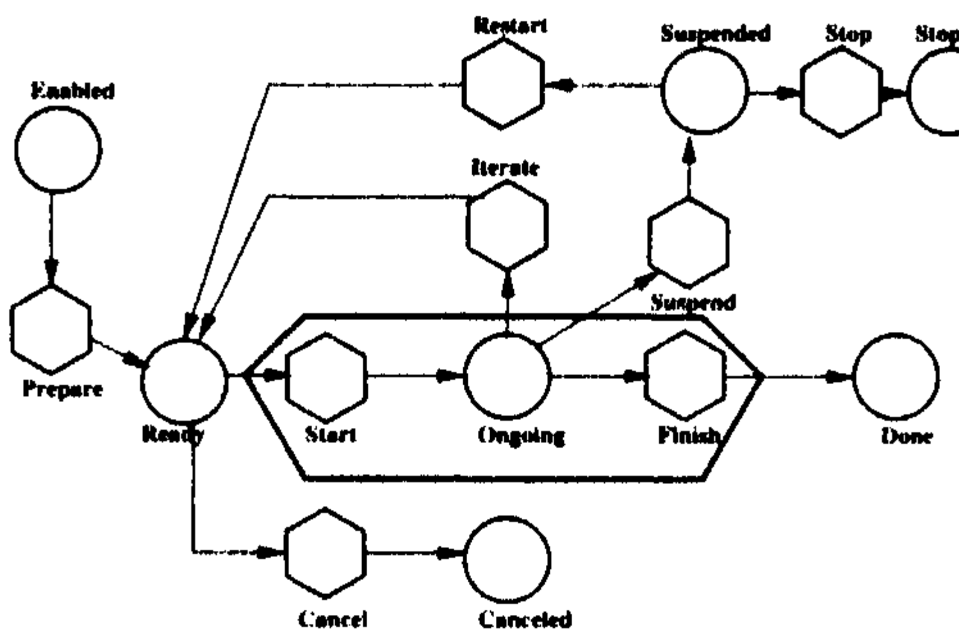

Figure 4: The CONTROLLER x-schema. Actions have rich internal structure that can be referred to by language and used for simulative inferences in language understanding.

(be + V-ing) which enable a speaker to focus on the ongoing nature of an underlying process while allowing for inferences that the process has started and that it has not yet completed. Similarly, one use of the *perfect* construction *(has V-ed)* allows a speaker to specify that-some *consequences* of the described situation hold. For instance, the phrase / have *lost mv kevs* entails that the keys are still missing (unlike the phrase *I lost my keys).* Languages also have a variety of other means to express aspect including aspectual verbs like *start, end, cease, continue,* and *stop* and related grammatical forms.

To model the kinds of subtle semantic distinctions made by languages, actions and events can no more be atomic transitions. In fact, we have found that cross-linguistically language makes reference to a specific structure of actions and events, which captures regularities that are relevant in the evolution of processes (enabling, inception, in-process, completion, suspension, resumption, etc.) We call this structure the CONTROLLER (Figure *4).* The controller abstraction seems to capture the basic temporal structure of people's conceptualization of events. The semantics of aspect arises from the dynamic binding between verb-specific x-schemas and a controller that captures regularities in the evolution of complex events, shown in Figure 1.

In our language understanding system, the causal domain structure is encoded as connected x-schemas. Our domain model is a dynamic system based on inter-x-sclierna activation, *inhibition* and *interruption.* In the simulation framework, whenever an executing x-schema makes a CONTROLLER transition, it potentially modifies state, leading to asynchronous and parallel triggering or inhibition of other x-schemas. The notion of state as a graph marking is inherently distributed over the network, so the working memory of an x-schema-based inference system is distributed over the entire set of x-schemas and state fluents. Of course, this is intended to model the massively parallel computation of the brain.

Figure 5 depicts a simplified x-schema model of walking and reacting to obstacles (the domain *of stumbling).*

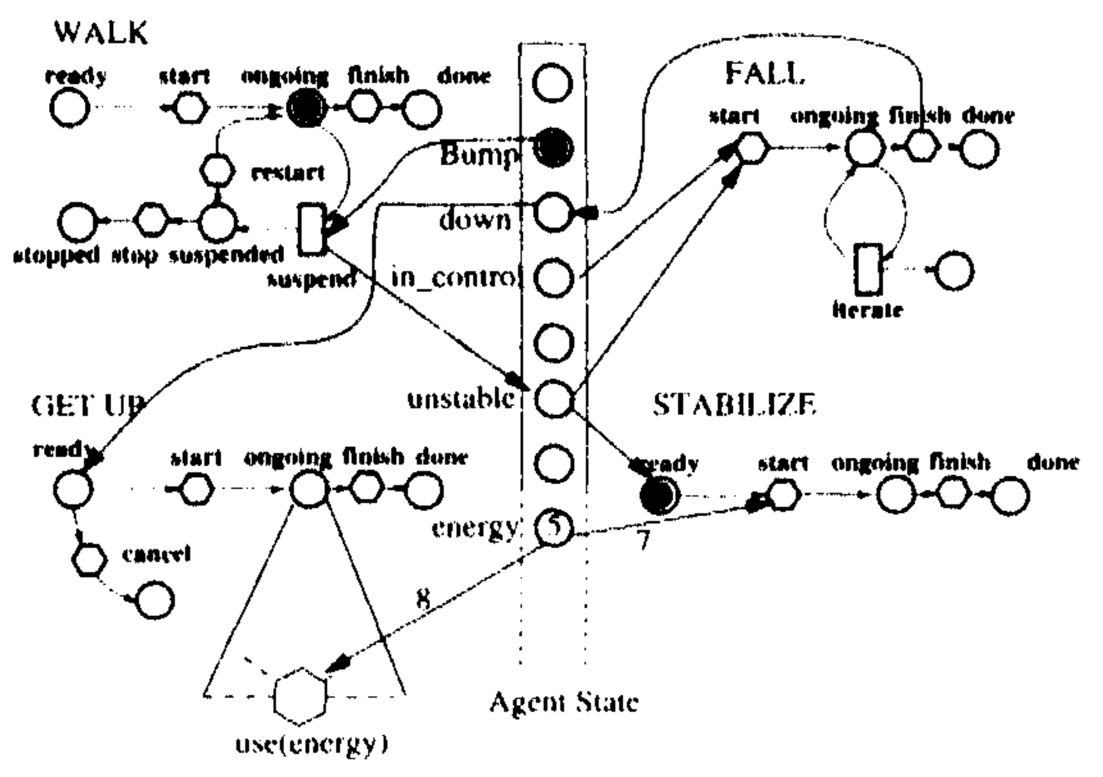

Figure 5: Event Structure is a x-schema simulation environment used for inference.

For instance, during a walk (specified by a token in the *ongoing* phase of the WALK x-schema) encountering an unanticipated bump, you become *unstable.* [4] This m*ay lead* to a FALL unless you are able to simultaneously *expend energy* and STABILIZE, in which case you may *resume* the *interrupted* walk. If you are unable to STABILIZE, and thus FALL, you will be down and hurt.

Now consider that this complex situation described above can be coded in a single lexical item *stumblel* First, notice that stumble can only occur during a STEP, and that it is a specific kind of interrupt to the step (i.e. the presence of a bump or stumbling block). But this by itself does not capture the intended meaning of stumble, since the inference (that the agent may fall) is routinely intended by the speaker. Furthermore, note that the fact that stumble is not a planned motion but an interrupt is important to infer that it is unintentional.

It should be clear that to model linguistic distinctions in event structure, we need much finer-grained distinctions than those that been proposed in the literature for reasoning about, actions. This is also consistent with the key observation in [MS, 1988] that aspectual phenomena depend on a notion of event structure that captures contingency relationships among events. Our framework of an active action semantics embodies a precise model of such inter-event contingency.

While our solutions to the problems of aspect are outside the scope of this paper, the following inter-schema activation, inhibition, and modification relationships are intended to give the reader an idea of the fine-grained nature of contingency relationships involved.

Definition 3 . Activation: Activation relationships between schemas correspond to the case where executing one schema causes the *enabling, start* or *continued execution* of another schema. We are able to distinguish *concurrent* from *sequential* activation.

[4]In fact, the simulation is of finer granularity in that it is during an ongoing STEP (subschema of WALK), that the interruption occurs. This is not shown to simplify exposition.

X **activates** Y (Act $(X, Y) : X, Y \in \mathcal{SS}$) if some subset p of places marked in the result state $P(X_r)$ of X ($p \subseteq P(X_r)$) *enable* the **start** transition of Y ($p \subseteq *T(Y_s)$). X **enables** Y if ($p * T(Y_s)$ i.e. $*T(Y_s) \subseteq P(X_r)$).

seq_enables(X, Y): done(X) $\wedge (p \in P(X_r)) \wedge (p \subseteq *T(Y_e^+))$
conc_enables(X, Y) : $ongoing(X) \wedge (p \in P(X_p)) \wedge (p \subseteq *T(Y_e^+))$
inh_periodic(X) : **seq_enables**(X, X)
mut_enables(X, Y) : **seq_enables**(X, Y) $\wedge$ **seq_enables**(Y,X)

Definition 4 . Inhibition: Inhibitory links prevent execution of the inhibited x-schema by activating an inhibitory arc. Again, our model is able to distinguish between concurrent and sequential inhibition as well as be able to model mutual inhibition and aperiodicity.

X inhibits Y (Inh $(X, Y) : X, Y \in \mathcal{SS}$) if some subset p of places marked in the *done* state of X $P(X_r)$ ($p \in P(X_r)$) are *inhibitor* arcs for the start transition of Y ($p \subseteq *T(Y_s)$). X disables Y if ($p * T(Y_s)$ i.e. $*Inh(Y_s) \subseteq P(X_r)$).

seq_disables(X, Y): done(X) $\wedge (p \in P(X_r)) \wedge (p \subseteq *T(Y_e^-))$
conc_disables(X, Y) : $ongoing(X) \wedge (p \in P(X_p)) \wedge (p \subseteq *T(Y_e^-))$
inh_aperiodic(X) : **seq_disables**(X, X)
mut_disables(X, Y) : **seq_disables**(X, Y) $\wedge$ **seq_disables(Y,X)**

Definitions 5 Modification: Modifying relationships between x-schemas occur when the execution of the modifying x-schema results in setting the Agent State in such a way the the currently active modified x-schema undergoes a controller state transition.

interrupts(X, Y): ongoing(Y) $\wedge (p \in P(X))) \wedge (p \supseteq *T(Y_{inh}^+))$
prevents(X, Y) : $enabled(Y) \wedge \neg start(Y) \wedge (p \in P(X_r)) \wedge (p \in *T(Y_s^-))$
terminates(X, Y) : $ongoing(Y) \wedge (p \in P(X_r)) \wedge (p \supseteq *T(Y_f^+))$
resumes(X, Y) : $suspended(Y) \wedge (p \in P(X_r)) \wedge (p \supseteq *T(Y_{int}^+))$
stops(X, Y) : $(suspended(Y) \vee ongoing(Y)) \wedge (p \in P(X_r)) \wedge (p \supseteq *T(Y_{cease}^+))$

Example 3 Examples of contingency relations in the Walking Domain

TRIP seq.enables FALL $\wedge$ STABILIZE.
-in_control(loc) enables FALL
-stable, enables STABILIZE
GRASPING(x) conc_enables HOLDING(x).
Energy(x) isa.resource for WALK

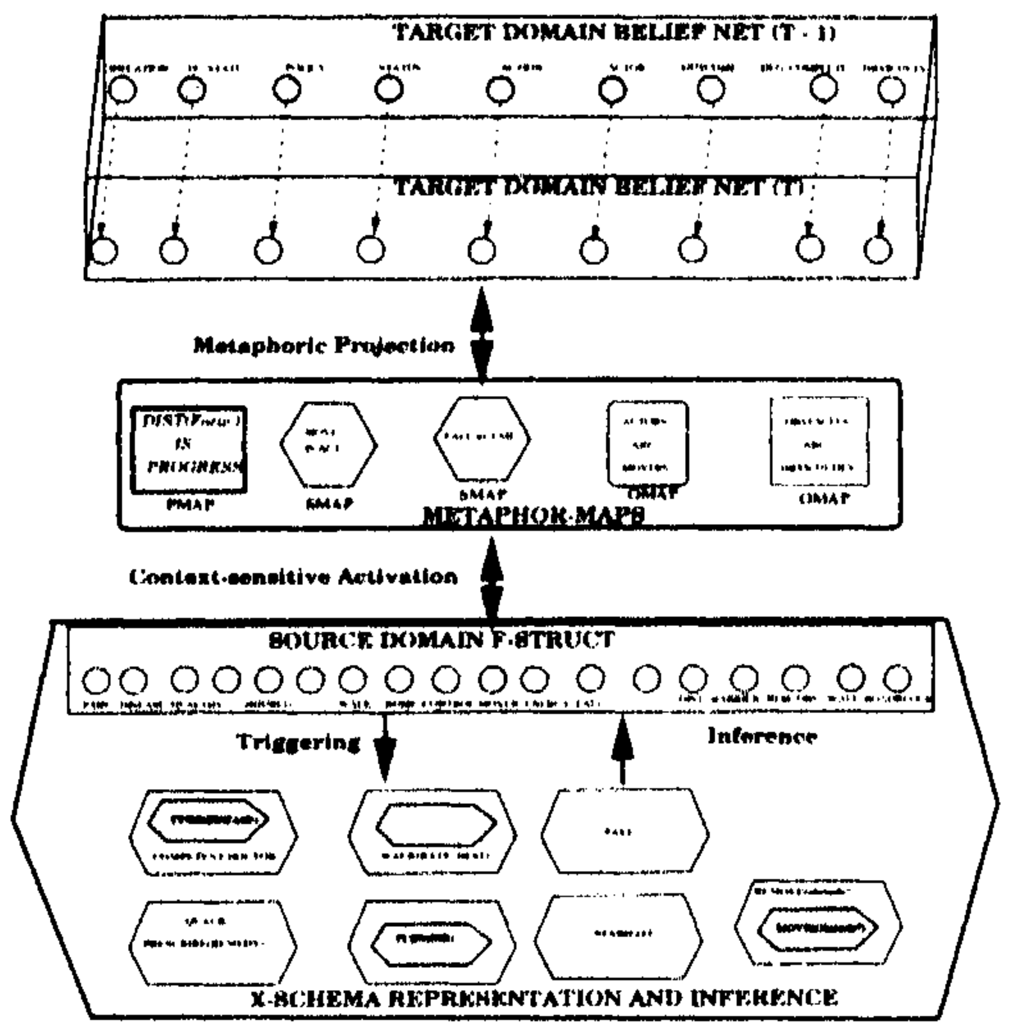
TARGET DOMAIN BELIEF NET (T - 1)
TARGET DOMAIN BELIEF NET (T)
Metaphoric Projection
DISTRESS IN PROGRESS
PMAP
SMAP
SMAP
OMAP
OMAP
METAPHOR-MAPS
Context-sensitive Activation
SOURCE DOMAIN F-STRUCT
Triggering
Inference
X-SCHEMA REPRESENTATION AND INFERENCE

Figure 6: Abstract Domains are mappings from concrete actions

WALK is mut-exclusive to RUN.
STABILIZE seq_disables FALL.
In_control(loc) disables FALL
GETUP resumes WALK.
Standing enables WALK
REACH(X) terminates WALK(X).
■

4.1 Metaphoric Reasoning about Actions and Events

We have seen how the structure of actions and events is grounded in fine-grained, dynamic representations. Another ubiquitous phenomenon in language[Lakolf, 1991], is the routine projection of such fine-grained semantic distinctions across domains. Systematic metaphors project features of these representations (source) onto abstract, domains such as economics (target) enabling linguistic devices to use embodied causal terms to describe features of abstract actions and processes.

Figure 6 shows an implemented system that uses projections of the action representation outlined earlier to interpret such sentences. In our model *indirect effects* of x-schema execution now not only propagate to dependent source domain fluents but may also be mapped by metaphor maps to other abstract domains (modeled as a temporally extended Relief Net,).

Continuing with the stumble example, notice that in our model effects of spatial inferences such as stumbling *leads to* falling can felicitously be transferred to the abstract, domain of economic policy through a conventionalized metaphor that, *falling ^ failure,* enabling the inference of plan failure. This inference context-sensitive and may be overridden by prior knowledge (in the target domain Relief Net) that the liberalization plan is succeeding.

5 Conclusion

This paper described a new framework for reasoning about actions that is motivated from story understanding. One central feature of this framework is an extremely fine-grained action model with a real-time execution semantics that is able to capture a much richer notion of contingency and causality than other models we are aware of. Another key feature is our model of state where complex dependencies between state variables are modeled as a Relief Network. We showed how this framework is able to reason about actions, ramifications with inertial and dependent fluents, as well as inter-domain mappings which are crucial in story understanding. We believe that looking further into issues in narrative (such as into force-dynamics, rnodals, and mental spaces) can yield valuable insights that can help us build useful theories of reasoning about actions.

6 Acknowledgments

Thanks to .1. Feldrnan, G. Lakoff and the NTL group at DC Berkeley and 1CSF

References

[Celfond *k* Lifschitz, 190:]] Celfond, M. & Lifschitz. V. (1993). Representing Action and Change by Logic Programs. *Journal of Logic Programming,* 17:301-322, 1993.

[Giunchiglia &. Lifschitz, 1995] Ciunchiglia F. & Lifschitz, V. (1995). Dependent, Fluents *Proc. Of IJCAI* 1995: 1964-1969, Morgan Kaufman, Inc. 1995.

Goldszmidt, 1992] Goldszmidt, M. *k* Pearl, J. (1992). Rank-based systems. *Proc. Of the Third Conference on Principles of KR and Reasoning,* 1992: 661-672, Morgan Kaufman, Inc. 1992.

[.Jensen, 1996] Jensen, F. (1906). *An Introduction to Bayesian Networks.* Springer-Verlag ISBN 0- 387-91502-8.

[Lakoff, 1994] Lakoff, G. (1994). What is Metaphor?. *Advances in Connectionist Theory. V3 : Analogical Connections,* V3,1994.

[McCarthy, 1969] McCarthy, .1., *k* Hayes, P. (1969). Some Philosophical Problems From the Standpoint of Artificial intellgence. *M a c h i n e J n t e Hi g e n ce* 4 (1!) 6 9) 463-502.

[Narayanan, 1997] Narayanan, S. (1997). Knowledge-based Action Representations for Metaphor and As pert (KARMA). *Ph.D. Dissertation* CS Division, FECS Dept, UC Berkeley 1997.

[MS, 1988] Moens, M. & Steedman, M. (1988). Temporal Ontology and Temporal Reference. In *Proc. ACL 88,* V4, Number 2, June 1988, pp. 15-29.

[Reisig, 1985] Reisig, W. (1985). *Petri Nets.* Springer Verlag.