

# Reasoning About Actions in Narrative Understanding

Srinivas Narayanan

CS Division, UC Berkeley and ICSI

snarayan@{cs,icsi}.berkeley.edu

## Abstract

Reasoning about actions has been a focus of interest in AI from the beginning and continues to receive attention. But 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, 1969], 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 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 infer-*

*ences* 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 cognitively motivated solutions to classical problems in logical theories of reasoning about actions. We first present the main features of our representation and show that it 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; Murata, 1989] called x-schemas, where the modifications allow for stochastic transitions, as well as special compositional states and transitions decomposable to an ordered collection of primitive states and transitions). 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* ( $T$ ) connected by weighted directed arcs  $\mathcal{A}$  ( $\mathcal{A} \in (P \times T) \cup (T \times P)$ ). Each arc  $a_{ij} \in \mathcal{A}$  has weight  $w_{ij}$  ( $w_{ij} \in \mathcal{N}$ ). Input Arcs  $*T$  ( $*T \in (P \times T)$ ) connect Input Places to Transitions. Output Arcs  $T*$  ( $T* \in (T \times P)$ ) connect Transitions to Output Places. Arcs are typed as *enable* arcs  $\mathcal{E}$ , *inhibitory* arcs  $\mathcal{I}$ , or *resource* arcs  $\mathcal{R}$ .

X-schemas have a well specified real-time execution semantics where the **next state** function is specified by the **firing rule**.<sup>1</sup> 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-schema** A transition  $T$  is said to be **enabled** if **no** *inhibitory* arc  $i \in \mathcal{I}$  of  $T$  has a **marked** source place **and** all sources of enable arcs  $e \in \mathcal{E}$  of  $T$  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  $P$  to  $T$ . The **firing** of an **enabled** transition  $T$ , removes  $w_{PT}$  tokens from the source of each non-inhibitory, non-enabled input arc  $P$  and places  $w_{TP}$  tokens in each output place of  $T$ .

X-schemas cleanly capture sequentiality, concurrency and event-based asynchronous control; with our extensions they also model *hierarchy*, *stochasticity* and *parameterization* (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 individualized 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 firing 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*, *dura-*

<sup>1</sup>Places are depicted graphically as circles, transitions as rectangles, standard arcs as directed edges, inhibitory arcs as undirected edges with unfilled circles, and tokens as filled circles or integers inside places.

Transition	Graphic	Firing Function
Stochastic		
Durative		
Instantaneous		
Hierarchical		Activate subnet Wait for return OR timeout

Figure 1: Basic types of transitions.

ive, instantaneous and hierarchical transitions. An instantaneous transition (shown as dark rectangles) *fires* 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 timeout.

**Theorem 1 . (proof sketch in Appendix A. Full proof in [Narayanan, 1997])** An x-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-markov 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, 1996; Pearl, 1988]. A belief network is a convenient data structure to encode causal domain knowledge. The basic algorithms that operate on the probabilistic network data structure deal both with new observations and database updates due to external intervention such as actions and random disturbances.

Belief networks consist of a set of variables and a set of directed links. Each variable has a finite set of mutually exclusive states. The variables together form a *DAG* (Directed Acyclic Graph). To each variable  $A$  with parents  $B_1 \dots B_n$  there is attached a conditional probability table  $P(A|B_1, \dots B_n)$ .<sup>2</sup> For a Belief Net  $B_u$ , the

<sup>2</sup>Note that the *CPT* entries could instead use the ranking function method of  $\kappa$ -calculus [Goldszmidt, 1992], where instead of measuring probabilities on a scale from zero to one, one imagines them being mapped onto a quantized logarithmic scale, and then treating beliefs mapping onto different quanta as being of different orders of magnitude.

following theorem allows us to calculate the joint probability  $P(U)$  from the conditional probabilities in the network.

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

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

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 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  $M_0$ . The next state function described earlier takes the system to a new marking  $M_{int_0}$ . 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 a posteriori assignment (*MAP*) of values to the state variables. The new state  $S_1$  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  $M_{int} = *a_i \cup M_i$ . 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 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 *ARD* [Giunchiglia & Lifschitz, 1995], we model both “inertial” (**always**  $C$  (where  $C$  is a formula)) and “dependent” ( $A$  **depends\_on**  $B$ ) fluents.

An *ARD* language consists of

1. A set of symbols  $\mathcal{F}$ , called *Fluent* names ( $\mathcal{F} \neq \{\}$ ), and another disjoint set of symbols  $\mathcal{A}$ , called *Action* names ( $\mathcal{A} \neq \{\}$ ) and  $\mathcal{A} \cap \mathcal{F} = \{\}$ . For every Fluent  $F$  there is a non-empty set  $Dom_F$  called the domain of  $F$ .
2. Two sets of symbols  $I, D : I \in \mathcal{F} \wedge D \in \mathcal{F}$ , where fluents in set  $I$  are called *Inertial* fluents and fluents in set  $D$  are called *Dependent* fluents. ( $I \cap D = \{\}$ ).

An *atomic formula* is of the form  $F$  **is**  $V$ , where  $V$  is in the domain of fluent  $F$ . If  $F$  is propositional, then  $Dom_F = \{true, false\}$ , and we write  $F$  for the formula  $F$  **is** *true*. A *value proposition* is of the form  $C$  **after**  $A^n$ , where  $C$  is a formula and  $A^n$  is a string of action names. If  $A^n$  is empty, we write **initially**  $C$ . A *constraint* is of the form **always**  $C$ , where  $C$  is a formula. An *effect proposition* is of the form  $A$  **causes**  $C$  **if**  $P$ , where  $A$  is an action, and  $C$  and  $P$  are formulas. A *dependency proposition* is of the form  $F$  **depends\_on**  $G$  **if**  $P$ , where  $F$  is the dependent fluent name,  $G$  any fluent name, and  $P$  a formula.

The following rules present the basic encoding of action theories (assuming the syntax of the *ARD* 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  $M_i(f) = 1$ .<sup>3</sup>
2. Domain Constraints with inertial fluents are modelled as instantaneous transitions. Statements in *ARD*, of the form **always**  $A \supset B$ , add an instantaneous transition with pre-set  $*T_{AB} = A$ , post-set  $T_{AB*} = A, B$ . 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).

<sup>3</sup>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.

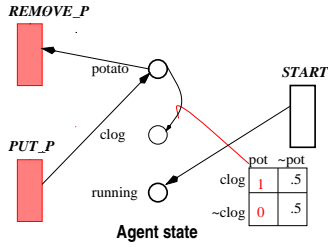


Figure 2: Potatoes in Tailpipes

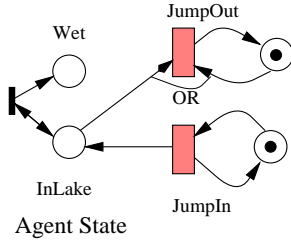


Figure 3: Jumping Into Lakes

3. **Initially**  $C$ , is modeled by assigning an initial marking where  $\forall f \in \mathcal{F} \cap C : M_0(f) = 1$ .  $\forall f_i, f_j \in \mathcal{F}$ , if  $f_i$  **depends\_on**  $f_j$ , add an instantaneous transition  $T_{\mathcal{I}\mathcal{J}}$  with preset  $*T_{\mathcal{I}\mathcal{J}} = f_j$  and post set  $T_{\mathcal{I}\mathcal{J}}^* = f_i, f_j$ .

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<sub>p</sub>*, *Remove<sub>p</sub>* (put/remove potato in/from tailpipe), *Start* (start the engine). The domain is characterized as *Put<sub>p</sub>* **causes** *potato*, *Remove<sub>p</sub>* **causes**  $\neg$ *potato* *Start* **causes** *running*, *clog* **depends\_on** *potato*, **always** *potato*  $\supset$  *clog*. ■

**Example 2 Wetness** Here the idea is that Jumping into a lake (*JumpIn*) 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 *ARD* as **always** *InLake*  $\supset$  *Wet*, *JumpIn* **causes** *InLake*, *JumpOut* **causes**  $\neg$ *InLake*, **Initially**  $\neg$ *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 *ARD*) 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  $A$ .

### 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 *JumpIn* 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<sub>pot</sub>*; *remove<sub>pot</sub>*, is not entailed by the description depicted in Figure 2 (from the uniform prior for  $P(\text{clog}|\neg\text{potato})$ ); if the domain theory contains the explicit knowledge **initially**  $\neg$ *clog*  $\wedge$   $\neg$ *potato*, the proposition is entailed by the model.<sup>4</sup>

## 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 [Vendler, 1967; Dowty, 1979; MS, 1988].

Many languages have a variety of grammatical aspectual modifiers such as the English *progressive* construction (*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 *I have lost my keys* 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 (en-

<sup>4</sup>More generally, logical entailment can be viewed as the downward closure of the final marking.

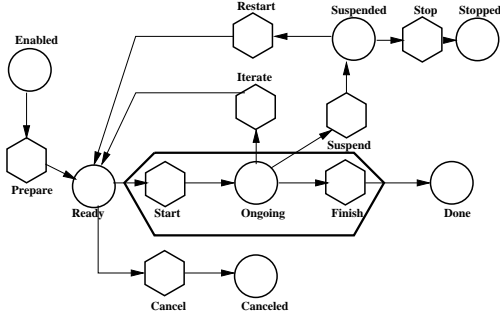


Figure 4: The CONTROLLER x-schema. Actions have rich internal structure that can be referred to by language and used for simulative inferences in language understanding.

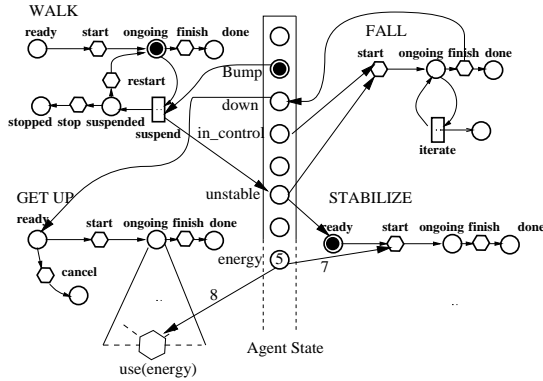


Figure 5: Event Structure is a x-schema simulation environment used for inference.

abling, 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 4.

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-schema *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*). 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*.<sup>5</sup> This may 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 *stumble!* 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.

$X$  **activates**  $Y$  ( $\text{Act}(X, Y) : X, Y \in \mathcal{SS}$ ) 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**( $X, Y$ ):  $\text{done}(X) \wedge (p \in P(X_r)) \wedge (p \subseteq *T(Y_e^+))$

**conc\_enables**( $X, Y$ ):  $\text{ongoing}(X) \wedge (p \in P(X_p)) \wedge (p \subseteq *T(Y_e^+))$

**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

<sup>5</sup>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.

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**( $Y, X$ )

**Definition 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.

$\neg in\_control(loc)$  enables FALL

$\neg stable$  enables STABILIZE

GRASPING( $x$ ) conc\_enables HOLDING( $x$ ).

$Energy(x)$  isa\_resource for WALK

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[Lakoff, 1994; Lakoff & Johnson, 1980], 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 embod-

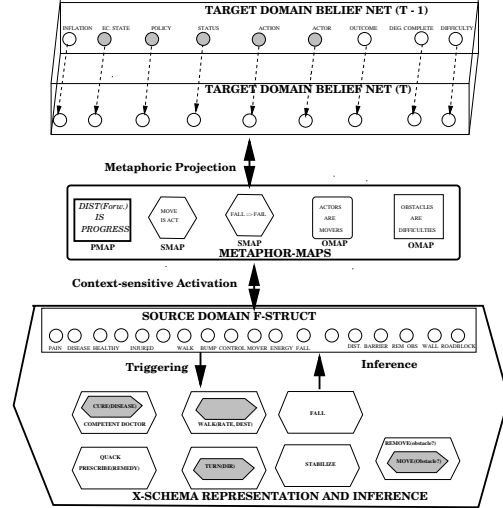


Figure 6: Abstract Domains are mappings from concrete actions

ied 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 Belief 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*  $\mapsto$  *failure*, enabling the inference of plan failure. This inference context-sensitive and may be overridden by prior knowledge (in the target domain Belief 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 Belief 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, modals, and mental spaces) can yield valuable insights that can help us build useful theories of reasoning about actions.

## 6 Acknowledgements

Thanks to Jerry Feldman, George Lakoff, Stuart Russell, Robert Wilensky and the NTL and RUGS groups at UC Berkeley and ICSI.

## References

- [Dowty, 1979] Dowty, D. 1979. *Word Meaning and Montague Grammar*. Synthese Language Library, Reidel, New York.
- [Gelfond & Lifschitz, 1993] Gelfond, M. & Lifschitz, V. (1993). Representing Action and Change by Logic Programs. *Journal of Logic Programming*, 17:301-322, 1993.
- [Giunchiglia & Lifschitz, 1995] Giunchiglia E. & Lifschitz, V. (1995). Dependent Fluents *Proc. Of IJCAI*, 1995: 1964-1969, Morgan Kaufman, Inc. 1995.
- [Goldszmidt, 1992] Goldszmidt, M. & 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. (1996). *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.
- [Lakoff & Johnson, 1980] Lakoff, G. and Johnson, M. (1980). *Metaphors we live by*. University Of Chicago Press.
- [McCarthy, 1969] McCarthy, J., & Hayes, P. (1969). Some Philosophical Problems From the Standpoint of Artificial Intelligence. *Machine Intelligence* 4 (1969) 463-502.
- [Murata, 1989] Murata, T. (1989). Petri Nets: Properties, Analysis, and Applications. In *Proc. IEEE-89, V77, Number 4, April 1989, pp. 541-576*.
- [Narayanan, 1997] Narayanan, S. (1997). Knowledge-based Action Representations for Metaphor and Aspect (KARMA). *Ph.D. Dissertation* CS Division, EECS 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*.
- [Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, San Mateo, Ca.
- [Reisig, 1985] Reisig, W. (1985). *Petri Nets*. Springer Verlag.
- [Vendler, 1967] Vendler, Z. (1967). *Linguistics in Philosophy*. Cornell University Press, Ithica, New York.

## 7 Appendix A

Here we prove the equivalence between x-schemas and Petri nets. In this report, we will not be concerned about the stochastic transitions, and hence we will only prove the first part of the theorem. We begin with the following definitions.

### Definition 6 . Petri Net

A Petri Net ( $PN$ ) is a 5-tuple  $PN = (P, T, I, O, M_0)$  where

- $P = \{p_1, \dots, p_n\}$  is a finite and non empty set of places,
- $T = \{t_1 \dots t_m\}$  is a finite and non-empty set of transitions,
- $P \cap T = 0$ ,
- $I : T \rightarrow P^\infty$  is the *input* function, a mapping from a transition to a bag of places,
- $O : T \rightarrow P^\infty$  is the *output* function, a mapping from a transition to a bag of places,
- $M_0$  is the initial net marking, a function from the set of places to the non-negative integers  $\mathcal{N}$ ,  $M_0 : P \rightarrow \mathcal{N}$ .

### Definition 7 . Enabled Transition

A transition  $t_j \in T$  in a Petri Net  $N = (P, T, I, O, M)$  is *enabled* if  $\forall p_i \in P$ ,  $M(p_i) \geq \#(p_i, I(t_j))$  where  $\#(p_i, I(t_j))$  represents the multiplicity of place  $P_i$  in bag  $I(t_j)$ .

### Definition 8 . Dynamic Behavior

The next-state function  $\delta : Z_+^n \times T \rightarrow Z_+^n \times T$  for a Petri net  $N = (P, T, I, O, M)$ ,  $|P| = n$ , with transition  $t_j \in T$  is defined if  $t_j$  is enabled. The next state is equal to  $M'$  where:

$$M'(P_i) = M(P_i) - \#(p_i, I(t_j)) + \#(p_i, O(t_j)) \forall p_i \in P \quad (2)$$

This can be extended to a sequence of transitions as follows

$$\begin{aligned} \delta(M, t_j \sigma) &= \delta(\delta(M, t_j), \sigma), \\ \delta(M, \lambda) &= M \end{aligned} \quad (3)$$

where  $\lambda$  represents the null sequence.

### Definition 9 . HLPN

A HLPN is a 6-tuple  $HLPN = (P, T, C, I, O, M_0)$  where

- $P, T$  are the places and transitions of an ordinary  $P - T$  net defined earlier.
- $C$  is a color function defined from  $P \cup T$  into finite sets.
- $I$  and  $O$  are forward and backward incidence functions defined on  $P \times T$  st.

- $I, O \in [C(t) \rightarrow C(p)_{MS}], \forall (p, t) \in P \times T$
- $M_0$  is a marking function defined on  $P$  describing the initial marking such that  $M_0(p) \in C(p)_{MS}, \forall p \in P$ .

**Definition 10 . Dynamic behavior of HLPN's.**

- A transition  $t \in T$  is *enabled* in a marking  $M$  w.r.t. a color  $c' \in C(t)$  (denoted by  $M(t, c') >$ , iff  $M(p)(c) \geq (p, t)(c')(c), \forall p \in P, c \in C(p)$ ).
- An enabled transition may *fire* in a marking  $M$  w.r.t. color  $c' : c' \in C$  yielding a new marking  $M'$  (denoted as  $M \rightarrow M'$ ) or  $M(t, c') > M'$  with:

$$M'(p)(c) = M(p)(c) + O(p, t)(c')(c) - I(p, t)(c')(c), \forall p \in P, c \in C(p). \quad (4)$$

- The various properties defined for  $P - T$  nets can also be defined for a *HLPN*. The reachability set of a HLPN  $R(HLPN) := R(HLPN, M_0) := \{M | M_0 \xrightarrow{*} M\}$  where  $\xrightarrow{*}$  is the reflexive and transitive closure of  $\rightarrow$ .

**Definition 11 . Bounded-net.**

Given a marked  $PN \Sigma = (P, T, C, I, O, M_0)$ , a place  $p \in P$  is *k-bound* iff  $\forall M_i \in R(N, M_0) (\#p \leq k)$ .  $\Sigma$  is *k-bounded* iff  $\forall p_i \in P$  ( $p_i$  is  $k_i$  bounded) and  $K = \max_i [k_i]$ .

**Lemma 1 . X-schemas (with finite parameter/values) can be converted to bounded HLPN's with finite number of colors.**

**Proof:** The proof proceeds in several steps. First, we note that nets with individual tokens (attribute-values) can be translated into colored Petri Nets. The reader is referred to [Murata, 1989] for the proof. We will now prove that x-schemas are translatable to bounded colored petri nets.

1. *enable* conditions (preconditions/etc.) consist of tokens that are not consumed at the firing of a transition. The incidence function is  $m_{p_o} - m_{p_i} + m_{p_{ei}}$ .
2. Preconditions and post-conditions are 1-bounded.
3. *consume* arcs correspond to the standard PN input arc.
4. Consumed resources are bounded from their initial values. In a finite number of transitions the net reaches the final marking. Since in each step the amount of produced resource is finite ( $\leq \max T^*$ ), the total number is finite as well.
5. From the discussion above, we conclude that without inhibitory arcs, the x-schemas are bounded ordinary Petri Nets. *Inhibit* arcs are a standard PN extension. With the introduction of inhibitor arcs, a transition is enabled when all of its non-inhibitory inputs  $I_{std}$  are marked with  $(I_{std}) \geq w_{stdT}$  and

places with inhibitory arcs onto  $T$  are empty. In general, introduction of inhibitory arcs increase the modeling power of Petri Nets to that of Turing machines; however in the case of bounded (*k-safe*) nets, a net with inhibitor arcs can always be transformed into an equivalent net without inhibitor arcs.

From the discussion above, we conclude that x-schemas can be encoded as Colored Petri Nets. ■

**Lemma 2 . HLPN's with finite number of colors can be unfolded into a unique ordinary net PN . Also Bounded HLPN  $\rightarrow$  Bounded PN.**

**Proof:** A *HLPN*  $= (P, T, C, I, O, M_0)$  can be *unfolded* into an ordinary Petri net  $PN$  in the following way:

1.  $\forall p \in P, c \in C(p)$ , we create a place  $(p, c)$  in  $PN$ .
2.  $\forall t \in T, c' \in C(t)$ , we create a transition  $(t, c')$  in  $PN$ .
3. We define the input ( $I$ ) and output ( $O$ ) function on  $PN$  as

$$I(p, c)(t, c') := I(p, t)(c')(c) \quad (5)$$

$$O(p, c)(t, c') := O(p, t)(c')(c) \quad (6)$$

4. The initial marking of  $PN$  is

$$M_0(p, c) := M_0(p)(c), \forall p \in P, c \in C(p) \quad (7)$$

The unfolded net  $PN$  is thus given by:

$$PN = \left( \bigcup_{p \in P} \bigcup_{c \in C(p)} (p, c), \bigcup_{t \in T} \bigcup_{c' \in C(t)} (t, c'), I, O, M_0 \right) \quad (8)$$

From the discussion above we conclude that x-schemas are equivalent to ordinary Petri nets. ■

## 7.1 The Stochastic Case

The continuous time stochastic Petri net was introduced by Molloy (Molloy 1982). As described above, the firing time is governed by an exponentially distributed random variable  $x_i$ . The firing time of transition  $t_i$  is given by

$$\mathcal{F}_{x_i} = 1 - e^{-\lambda_i x} \quad (9)$$

The negative exponential distribution renders the reachability graph of the *SPN* isomorphic to a continuous

time Markov chain. The Markov chain  $MC$  can be obtained from the reachability graph as follows: The  $MC$  state space is the reachability set  $R(PN)$  of the marked  $SPN$ . In  $MC$ , the transition rate from  $M_i$  to  $M_j$  is given by  $q_{ij} = \lambda_k$ , corresponding to the firing rate of the transition  $t_k$  from  $M_i$  to  $M_j$ . If several transitions lead from  $M_i$  to  $M_j$ , then  $q_{ij}$  is the sum of the rates of these transitions. If there is no link from  $M_i$  to  $M_j$  in  $R(PN)$  then  $q_{ij} = 0$  in  $MC$ .

The steady state distribution  $\pi$  of the  $MC$  is obtained by solving the linear equations:

$$\begin{aligned} \pi Q &= 0 \\ \sum_{i=1}^s \pi_i &= 1 \end{aligned} \quad (10)$$

$$(11)$$

From the vector  $\pi = (\pi_1, \dots, \pi_s)$  we can compute the following measures:

- **Probability of being in a set of states:** Let  $\mathcal{B} \subseteq R(PN)$  constitute the states of interest in a given  $SPN$ . Then the probability of being in a state of the corresponding  $SPN$  is

$$P(\mathcal{B}) = \sum_{M_i \in \mathcal{B}} \pi_i \quad (12)$$

- **Probability of taking a transition  $t_j$ :** Let  $EN_j$  be the subset of  $R(PN)$  in which the transition  $t_j$  is enabled. Then the probability that an observer looking randomly into the next sees  $t_j$  firing next ( $p_j$ ) is given by

$$p_j = \sum_{M_i \in EN_j} \pi_i \frac{\lambda_j}{-q_{ii}} \quad (13)$$

where  $q_{ii}$  is the sum of the transition rates out of  $M_i$ .

- The throughput of a transition is the mean number of firings at steady state

$$d_j = \sum_{M_i \in EN_j} \pi_i \lambda_j \quad (14)$$

**Definition 1** *HLSPN*: A *HLSPN* is a 3-tuple  $HLSPN = (HLPN, T, W)$  where

$HLPN = (P, T, C, I, O, M_0)$  is the underlying marked high-level Petri net.

$T$  is a set of timed transitions.

$W = (w_1 \dots w_{|T|})(w_i \in \mathcal{R})$  is an array whose entries are negative exponential distributions specifying the firing delay. This rate could be marking dependent.

**Definition 12** . *HLGSPN*: A *HLGSPN* is a 4-tuple  $HLGSPN = (HLPN, T_1, T_2, W)$  where

- $HLPN = (P, T, C, I, O, M_0)$  is the underlying marked High-Level Petri net.
- $T_1 \subseteq T$  is a set of timed transitions  $T_1 \neq \emptyset$ .
- $T_2 \subset T$  is a set of immediate transitions.
- $T_1 \cup T_2 = T$  and  $T_1 \cap T_2 = \emptyset$ .
- $W = (w_1 \dots w_{|T|})(w_i \in \mathcal{R})$  is an array whose entries

1. Could be a negative exponential distribution specifying the firing delay when  $t \in T_1$ . For such a timed transition, the rate could be marking dependent.
2. Is a firing weight for the immediate transition  $t$ , when  $t \in T_2$ . This weight is also possibly marking dependent.

If  $T_2 = \emptyset$ , then the *GSPN* defined above essentially reduces to an *SPN*. Hence  $SPN \subset GSPN$ . The presence of immediate transitions makes the reachability graph of a marked *GSPN* non-markovian since immediate transitions fire in zero time. Markings where only immediate transitions are enabled are called **vanishing** since an external observer will never see these states, even though the stochastic process sometimes visits them. For timed transitions, the stochastic process sojourns for an exponentially distributed amount of time, and states resulting from markings enabling timed transitions are likely to be seen by an external observer. These states or markings are called **tangible** markings.

### Lemma 3 . X-schema to HLGSPN

*X-schemas with stochastic, hierarchical, instantaneous and durative transitions are isomorphic to HLGSPNs.*

**Proof (sketch):** *The proof is simple, since the stochastic and immediate transitions correspond to the GSPN transitions. Hierarchical transitions can be modeled using a pair of instantaneous transitions (corresponding to in and out of the subnet) with a deterministic (or stochastic) timeout transition.* ■

### Lemma 4 . HLGSPN to GSPN

*HLGSPNs can be converted to Generalized Stochastic Petri Nets (GSPN).*

**Proof (sketch):** *The proof is exactly similar to the non-stochastic case, since the typing of tokens does not depend on the types of transitions.* ■

From Lemma 1 and Lemma 3 we conclude that x-schemas can be encoded as Generalized Stochastic Petri Nets (*GSPN*), thereby proving Theorem 1.