M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 456
January 1998

Fast Constraint Propagation on Specialized Allen Networks and its Application to Action Recognition and Control

Claudio Pinhanez **Aaron Bobick**
20 Ames St. – Cambridge, MA 02139
pinhanez | bobick@media.mit.edu

Abstract

In this paper we present a specialization of Allen interval networks that permits the rapid determination as to whether a given interval must be occurring at the current point in time. The Allen-closure of the interval network is projected into a 3-valued (past,now,fut) constraint network called a PNF-*network*. We show that the minimal domain of a PNF-network can be approximately computed in linear time by using arc-consistency. This computation is the key factor in the PNF *propagation* method of determining, for each instant of time and given information from perceptual sensors, the PNF-state of each interval, that is, happening (now), already happened (past), or has not happened (fut). We show how the computation of PNF-states can support both action recognition and the control of real-time interactive environments in which the actions are described by Allen interval networks.

1 Introduction

Unlike previous work on constraint propagation in interval algebra networks [22, 5, 12], we are not interested in the computation of a set of time intervals for each node that is compatible with the temporal constraints. Instead, this work is focused on the determination of whether an interval is happening (now), already happened (past), or has not happened (fut) — what we call the PNF-*state* of the interval. With this simple information about the intervals of a network we show that is possible to perform human action recognition and control of interactive systems whose temporal structure is described using Allen's temporal relationships [1].

Most algorithms for temporal constraint propagation in interval algebra networks are *NP-hard* [23]. In this paper we describe a method to circumvent this problem that is based on the projection of the Allen interval network into the specialized 3-valued (past,now,fut) constraint network called a PNF-*network*. The first section of the paper defines those constraint networks and shows that it is possible to approximately compute the minimal domain in linear time by using arc-consistency. Considering that the minimal domain contains all possible solutions for the network, we can determine the state of an interval just by verifying if the minimal domain is composed exclusively by one of the three symbols past,now, or fut.

However, computing the minimal domain of a PNF-network only produces the PNF-states of the intervals for a particular instant of time. Considering that the applications we are interested in typically require on-line processing, it is necessary to design a method which is able to incorporate information from past actions and sensed activities and their relationships with the current state of the sensors. Only in the presence of this information can we fully exploit the temporal constraints described by the interval algebra network. The second section of the paper describes PNF *propagation*, a method that determines the current PNF-state of the intervals considering information from perceptual sensors and the previous state of the intervals. PNF propagation represents sensor values and previous state as unary constraints on the PNF-network and, by computing the minimal domain, is able to rapidly determine the current PNF-state of the intervals in the network.

Although conservative, the PNF propagation approach is sufficiently strong to be used in practical, real-time applications. The first application described in this paper refers to the recognition of complex human actions described by an interval algebra network. The second application is our work on *interval scripts*, a paradigm for scripting interaction especially suit to immersive, action-driven computerized environments. In both cases we believe that PNF propagation provides an interesting compromise between the expressiveness of Allen's temporal relationships and the need for fast computation.

2 PNF-Networks

An *interval algebra network*, or simply an *IA-network* (also called by many authors as an *Allen interval network*), is a constraint satisfaction network [8] where the variables correspond to time intervals, and the arcs to binary temporal constraints between the intervals expressed using Allen's temporal relationships [1]. Allen's algebra employs disjunctions of the 13 possible primitive relationships between two time intervals: the relations *equal* (e), *before* (b), *meet* (m), *overlap* (o), *during* (d), *start* (s), *finish* (f), and their inverses, *ib*, *im*, *io*, *id*, *is*, and *if* (see [1] for a better definition of those relationships).

IA-networks are normally used in tasks involving planning or scheduling. Using constraint propagation techniques [12] it is *NP-hard* to determine if there are *solutions* to the network, i.e., an assignment of a time intervals to each variable satisfying all constraints. Given a variable of an IA-network, a *feasible value* for the variable is a time interval which belongs to at least one solution. The set of all feasible values is called the *minimal domain* of the variable. Unfortunately, calculating the minimal domain in IA-networks is also *NP-hard* in the number of constraints [23], preventing their use in most practical problems. This motivated us to search for specializations of IA-networks that can be solved in polynomial time but still handle our particular applications.

2.1 PNF-Network: Definition

A *PNF-network* is a 3-valued binary constraint satisfaction network where the domain of the all variables $w_1, w_2, \ldots, w_n$ is the set of symbols $m = \{\mathtt{past}, \mathtt{now}, \mathtt{fut}\}$. To simplify the notation, we define the set M of subsets of m,

$$\begin{aligned} M &= \{\emptyset, \{\mathtt{past}\}, \{\mathtt{now}\}, \{\mathtt{fut}\}, \{\mathtt{past}, \mathtt{now}\}, \\ &\quad \{\mathtt{past}, \mathtt{fut}\}, \{\mathtt{now}, \mathtt{fut}\}, \{\mathtt{past}, \mathtt{now}, \mathtt{fut}\}\} \end{aligned}$$

whose elements are abbreviated as

$$M = \{\mathtt{EMP}, \mathtt{P}, \mathtt{N}, \mathtt{F}, \mathtt{PN}, \mathtt{PF}, \mathtt{NF}, \mathtt{PNF}\}$$

An unary constraint P_i on a variable w_i can be represented by an element of M. For instance, $w_i \in \mathtt{PN}$ constrains w_i to be either past or now. We notate this by saying that $W_i = \mathtt{PN}$ is the *restricted domain* of w_i.

A binary constraint P_{ij} between two variables w_i and w_j is a truth matrix which determines which the admissible values are for the pair of variables (w_i, w_j). A compact way to represent the matrix is to use a triad $\langle r_P, r_N, r_F \rangle \in M \times M \times M$ where r_P represents the admissible values for w_j when $w_i = \mathtt{past}$, and similarly for r_N and r_F. For example, consider the binary constraint represented by $P_{ij} = \langle \mathtt{PN}, \mathtt{F}, \mathtt{F} \rangle$. According to this constraint, the only values that the pair of variables (w_i, w_j) can assume are (past, past), (past, now), (now, fut), and (fut, fut).

2.2 Projecting IA-networks into PNF-networks

Binary constraints between variables in a IA-network can be mapped into binary constraints in a PNF-network. Suppose a pair of variables (w_i, w_j) is constrained in the original IA-network by the relation *meet* ($\{m\}$), that is, the interval w_i is immediately followed by w_j. Intuitively, if w_i is happening now, $w_i = \mathtt{now}$, then w_j can only occur in the future, yielding $w_j = \mathtt{fut}$. Similarly, if $w_i = \mathtt{fut}$, then w_j must also be fut. Finally, if it is know that $w_i = \mathtt{past}$, the interval w_j must have already started, although it may have finished or not — $w_j \in \mathtt{PN}$.

As we saw above, the relationship *meet* ($\{m\}$) between w_i and w_j can be mapped into the binary constraint $P_{ij} = \langle \mathtt{PN}, \mathtt{F}, \mathtt{F} \rangle$ such as to preserve the temporal ordering contained in the original IA-network. Table 1 displays the function γ which maps the 13 Allen's primitive relationships into their equivalent binary-constraints between PNF symbols.

Given a binary constraint in an IA-network, i.e., a disjunction of primitive relationships Q, we can define its projection by the function Γ:

$$\Gamma(Q) = \bigcup_{\lambda \in Q} \gamma(\lambda)$$

where the union of the primitive binary constraints is defined as their component-wise union. For instance, if $Q = \{b, ib\}$, then

$$\begin{aligned} \Gamma(\{b, ib\}) &= \langle \mathtt{PNF}, \mathtt{F}, \mathtt{F} \rangle \cup \langle \mathtt{P}, \mathtt{P}, \mathtt{PNF} \rangle \\ &= \langle \mathtt{PNF}, \mathtt{PF}, \mathtt{PNF} \rangle \end{aligned}$$

$\gamma(Q) = \langle r_P, r_N, r_F \rangle$	r_P	r_N	r_F
e	P	N	F
b	PNF	F	F
ib	P	P	PNF
m	PN	F	F
im	P	P	NF
o	PN	NF	F
io	P	PN	NF
s	PN	N	F
is	P	PN	F
d	PN	N	NF
id	P	PNF	F
f	P	N	NF
if	P	NF	F

Table 1: The function γ which maps Allen's primitives into PNF-network constraints.

Given an IA-network, we can project it into a PNF-network where each binary constraint is projected into its PNF-equivalent through Γ.

2.3 Component Domains

As we will see below, most of our methods are related to the computation of the minimal domain of a PNF-network subjected to unary constraints. This motivated us to have an unified representation for domains and unary constraints that we call a component domain. Formally, given a PNF-network on variables $w_1, w_2, \ldots, w_n$, a *component domain* W of the PNF-network is any combination of restricted domains on the variables w_i, notated by $W = (W_i)_i = (W_1, W_2, \ldots, W_n)$, where each W_i is a subset of $\{\mathtt{past}, \mathtt{now}, \mathtt{fut}\}$.

The value of each W_i in a component domain W is called the PNF-*domain* of the variable w_i. We also denote by U the set of all component domains of a PNF-network, $U = M^n$, where n is the number of variables corresponding to the intervals in the PNF-network.

2.4 PNF-Restriction

Given a PNF-network and a component domain W, consider the problem of finding the minimal domain of the network by imposing the components of W as unary constraints on the network's variables. The result of the computation — also representable by a component domain — is called the *restriction of* W, $\mathcal{R}(W)$, where each component $\mathcal{R}(W)_i$ is the minimal domain of the variable w_i.

Computing the minimal domain in constraint satisfaction networks is, in general, a *NP-hard* problem [22]. In our experiments, we have been employing an *arc-consistency* algorithm (based on [10]) to compute an approximation of the the minimal domain $\mathcal{R}(W)$.

It is easy to show that the result of the arc-consistency algorithm is always a conservative approximation to the minimal domain; that is, it contains all the feasible solutions of a constraint satisfaction network. However, in our use of that algorithm (including many large scale runs), we have never found a situation where the arc-consistency algorithm has produced a component domain larger than the minimal domain. In other words, we have experimental reasons to believe that computing the minimal domain of a PNF-network is in fact a simpler problem than the computing of the minimal domain of a general constraint satisfaction network, and achievable in linear time by the arc-consistency algorithm. The constraint satisfaction network literature lists many cases where special characteristics of the network reduce the complexity of the problem [5, 12]. Presently we are trying to determine whether this conjecture is true.

3 PNF Propagation

PNF-restriction deals exclusively with determining feasible options of an action *at a given moment of time.* However, information from the previous time step can be used to constrain the occurrence of intervals in the next instant. For example, after an interval is determined to be in the past, it should be impossible for it to assume another PNF-value, since, in our semantics, the corresponding action is over. Similarly, if the current value of the interval is now, in the next instant of time it can still be now, or the corresponding action might have ended, when the interval should be past. To capture these ideas, we define a function that time-expands a component domain into another that contains all the possible PNF-values in the next instant of time.

3.1 Time Expansion

We define a *time expansion function*, $\mathcal{T} : U \to U$, that considers a component domain W^t at time t and computes the smallest component domain W^{t+1} at time $t+1$ that still satisfies the intuitive meanings of past, now, and future. We start by defining a time expansion function for each element of $m = \{\texttt{past}, \texttt{now}, \texttt{fut}\}$, $\mathcal{T}_m : m \to M$. A natural choice [1] is the time expansion function $\mathcal{T}_m : m \to M$

$$\mathcal{T}_m(\texttt{past}) = \texttt{P} \quad \mathcal{T}_m(\texttt{now}) = \texttt{PN} \quad \mathcal{T}_m(\texttt{fut}) = \texttt{NF}$$

Given the function that time-expands elements $\mathcal{T}_m$, we define the function that expands the elements of M, $\mathcal{T}_M : M \to M$ as being the union of the results of $\mathcal{T}_m$,

$$\mathcal{T}_M(\Phi) = \bigcup_{\phi \in \Phi} \mathcal{T}_m(\phi)$$

and the time expansion of a component domain W, $\mathcal{T} : U \to U$, as the component-wise application of $\mathcal{T}_M$ on a component domain $W = (W_i)_i$, $\mathcal{T}(W) = (\mathcal{T}_M(W_1), \mathcal{T}_M(W_2), \ldots, \mathcal{T}_M(W_n))$.

3.2 PNF Propagation: Definition

PNF *propagation* is a method to propagate temporal constraints through a sequence of events that considers available information about the current state and the influence of all the past states of the system.

Formally, given a PNF-network, PNF propagation is the process that determines the minimal domain of each interval at each time t, represented by the component domain W^t, called the *state of the* PNF*-network* at time t. Current state information is gathered in the component domain S^t where all states are PNF except those corresponding to actually detected values. Typically, the current state comes from sensors accessing directly the world or environment.

The initial component domain W^0 represents an initial state of total ignorance, $W^0 = (\texttt{PNF})_i$. After that, we determine W^t by time-expanding the previous state W^{t-1} — and therefore determining the possible sequences for that state —, intersecting it to the current state information S^t, and applying restriction to remove logical impossibilities due to temporal constraints. That is

$$W^t = \mathcal{R}(\mathcal{T}(W^{t-1}) \cap S^t)$$

It is necessary to time expand the component W^{t-1} before intersecting it with the perceptual information S^t, since between instant $t-1$ and t actions may have ended or begun.

4 PNF Propagation in Action Recognition

The first application in which we used PNF propagation is the recognition of human actions. The problem here is to determine whether given actions are occurring given the sequence of perceptual sensor values through time. Probabilistic methods for visual action recognition have been proposed in the computer vision community [3]. Howeve, we believe that it is important to exploit the fact that logical impossibilities prevent the occurrence of some sequences of sub-actions, and that it is necessary to incorporate such constraints into vision systems designed to recognize human action.

Some work in the vision community [9, 13, 21, 7] has attempted to incorporate logical definitions of action into perceptual mechanisms. However, these systems are unable to cope with most of the complex time patterns of everyday actions which include external events, simultaneous activities, multiple sequencing possibilities, and mutually exclusive intervals [2]. For example, Kuniyoshi and Inoue [9] use finite automata to represent actions performing simple actions to teach a robot. Implicit in the model is the assumption of strictly sequential sub-actions, which, although adequate for the mapping into robot primitives, seems to be a strong restriction for the representation of generic human actions.

Similarly, Nagel [13] uses *transition diagrams* to represent driver's maneuvers in a highway but provides no means to represent overlapping activities. Siskind's approach [20, 21] uses an *event logic* to represent basic actions, temporarily connected using Allen's primitive relationships. However, besides being a fully exponential modal logic, the proposed temporal propagation method using *spanning intervals* is quite inefficient.

Our proposal is to represent the temporal structure of actions and their sub-actions using the full expressiveness of an IA-network, but avoid the exponential computing time by projecting them into PNF-networks, and performing recognition using PNF propagation.

Allen's algebra was chosen as the underlying temporal formalism because we consider essential the ability to express mutually exclusive actions; for this reason, we can not use temporal algebras based on relations between endpoints, e.g. [5, 24], in spite of the better performance of the reasoning methods they provide.

[1] $\mathcal{T}_m$ assumes that the updating sampling rate is fast enough to assure that an interval can not go from fut to past without passing through the now state.

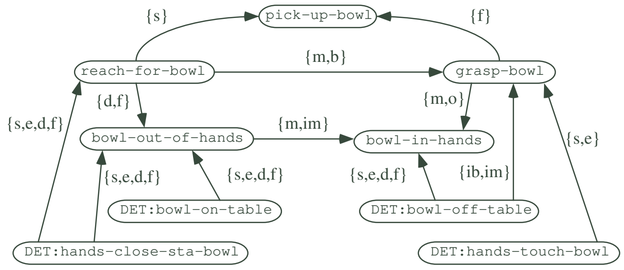

Figure 1: IA-network corresponding to the temporal structure of a "pick-up bowl" action .

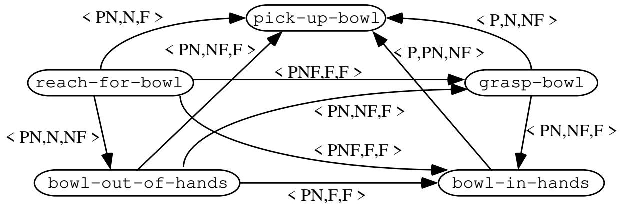

Figure 2: Projection of the closure of the IA-network shown in fig. 1 into a PNF-network. The domain of all node variables is PNF.

4.1 Representation of the Temporal Structure of Actions

In our approach, temporal structures of actions are represented by IA-networks. An example is show in fig. 1 that depicts the temporal structure of a "pick-up bowl" action. The interval `pick-up-bowl` corresponds to the period where the action of picking up the bowl is occuring. This action is decomposed into two sub-actions corresponding to the intervals `reach-for-bowl` and `grasp-bowl`.

The relations between those three intervals are defined by the arcs connecting them. The sub-action `reach-for-bowl` is declared to be a time interval that has the same beginning as `pick-up-bowl`, but finishes first — the $\{s\}$ relationship. Similarly, `grasp-bowl` finishes at the same time as `pick-up-bowl` ($\{f\}$). The relationship between `reach-for-bowl` and `grasp-bowl` is defined in more vague terms: they either immediately follow each other or happen in sequence — represented by the disjunction $\{m, b\}$. Implicit in this constraint is the fact that the two sub-actions are mutually exclusive.

The next level of decomposition involves two complementarity predicates (written as $\{m, im\}$) about the physical relation between the bowl and the hands, `bowl-in-hands` and `bowl-out-of-hands`. The fact that reaching for the bowl must happen while the bowl is not in contact with the hands is expressed by the $\{d, f\}$ relationship between `reach-for-bowl` and `bowl-out-of-hands` . Similarly, `bow-in-hands` starts during `grasp-bowl` or just after its end ($\{m, o\}$).

For each of the previous "physical" predicates we can assign simple, low-level detectors (marked by the prefix `DET:`). In the bottom layer, `DET:hands-close-sta-bowl` detects if the hands are close to the bowl while the bowl is static and on the table. `DET:hands-touch-bowl` fires only when the hand is touching the bowl. The other two detectors, `DET:bowl-on-table` and `DET:bowl-off-table`, identify the presence of the bowl on or off the table.

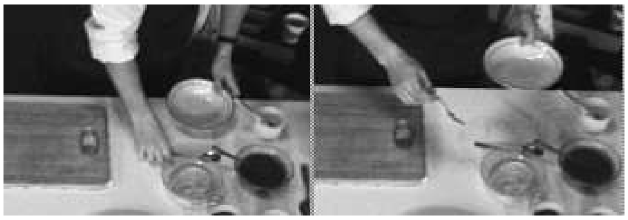

Figure 3: Images from the video used in the experiments.

By using Allen's representation system it is also possible to infer stronger temporal constraints between the actions by pre-processing the IA-network using Allen's path-consistency algorithm [1]. The result of the algorithm is an approximation of the *temporal closure* of the network, that is, the set of all temporal relations which logically follow from the initial constraints [24].

After the approximate closure is computed, the IA-network is projected into a PNF-network using the algorithm described above. Figure 2 shows the projection of the IA-network for "pick-up bowl" of fig. 1 into a PNF-network (the detector nodes are omitted for clarity).

4.2 Representation of Sensor Information

In our action detection method, sensor information is represented as unary constraints on the PNF-network by a component domain. Typically, we associate with a binary sensor only two values: N if the sensor is on, and PF if the sensor is off. Therefore, we can represent all the information coming from sensors by a component domain $S = (S_i)_i$, where

$$S_i = \begin{cases} \text{value of the } w_i\text{–sensor} & \text{if } w_i \text{ is a sensor} \\ \texttt{PNF} & \text{otherwise} \end{cases}$$

4.3 Results in Action Detection

We have been running experiments on three different actions, "pick-up bowl", "wrapping chicken", and "mixing ingredients". The temporal structure of each action is codified manually into an IA-network. The IA-networks are pre-processed using Allen's path-consistency algorithm (as revised in [24]) and then converted into PNF-networks.

To test the PNF formulation, we manually extract the values for the sensors by watching a video of the action being performed (see fig. 3). The sensed primitives are simple to detect using current computer vision technology; in fact, some of the sequences have been used before in a project involving tracking of people and objects in the domain of a cooking show [14]. We also determine the interval where every action and sub-action is actually happening and use the information to evaluate the performance of the action detection method.

In this paper we presume perfect sensors; a fault-tolerant extension of these ideas is described in [15]. We start by examining the results for the detection of the action "pick-up bowl" shown in fig. 4. The top part of the figure displays the temporal diagrams for the PNF-domains of the detectors (marked as `DET:`) and the true state of all other intervals (marked as `TRUE:`) for a particular instance of the action of

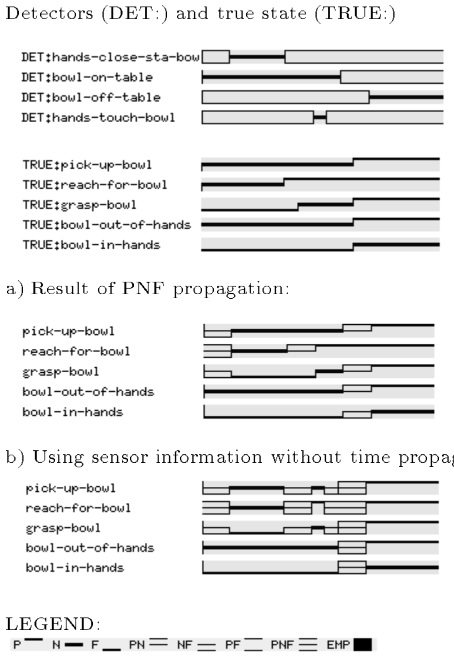

Figure 4: The detection of the action "pick-up bowl".

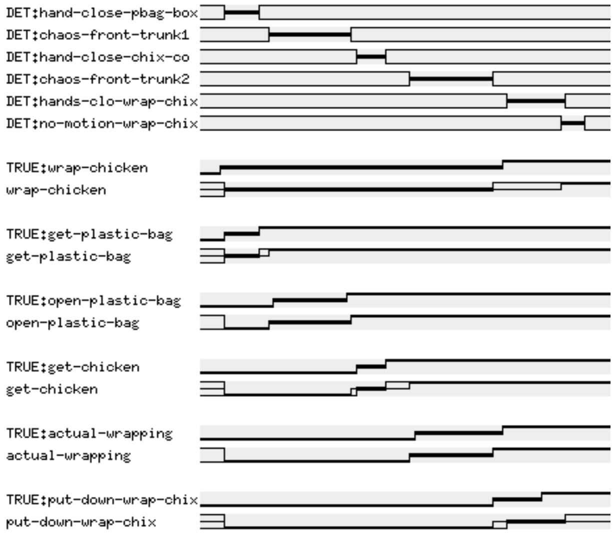

Figure 5: Detection of the action "wrapping chicken".

picking up a bowl. The data were obtained manually from a video depicting the action. The diagram employs different symbols for each `PNF`-domain, under the convention that the bottom line represents the `fut` state, the middle represents `now`, and the top represents `past` (see the legend at the bottom of fig. 4).

Figure 4a shows that occurences of the action "pick-up bowl" and its sub-actions are correctly determined most of the time (compare to the `TRUE:` diagram at the top of the figure). In [15] this example is more extensively analyzed in a discussion about the effects caused by the different temporal relationships between the actions.

Figure 4b shows the importance of the information from the previous instant of time on the strength of `PNF` propagation. In this case, W^t is computed solely on the information from the sensors, $W^t = \mathcal{R}(S^t)$. Comparing part b of fig. 4 with part a, we can see a distinct degradation in the results. The main reason is that after a cause for an interval being in the `now` state ceases to exist, the system still considers that the interval can happen in the `fut` (compare `pick-up-bowl` in both cases).

Figure 5 illustrates the detection of a more complex action, wrapping chicken with a plastic bag, which involves 25 sub-actions and 6 sensors. The constraints were designed to resemble the result of an automatic inference system operating on a compact description of the action based on Schank's conceptualizations [19]. In [18], Rieger discusses the implementation of such an inference system; in a previous work we have implemented a simpler system able to operate in a specific context of a cooking show domain [14].

Figure 5 displays the true and the recognized state of the main action and of five sub-actions, each of them with a level of complexity similar to the "pick-up bowl" shown above. All the sensors are very simple: proximity between hands and the box containing plastic bags (`DET:hand-close-pbag-box`) and the plate containing chicken (`DET:hand-close-chix-co`), chaotic movement in front of the subject's trunk (`DET:chaos-front-trunk`), and absence of motion in the area of the wrapped chicken (`DET:no-motion-wrap-chix`). Notice that the main action and the five sub-actions in the example are correctly detected most of the time.

5 Using `PNF` Propagation for Interaction Control

As shown in the action detection case described above, the `PNF` propagation method is capable of determining the `PNF`-state at each moment of time of a `PNF`-network, given the value of the sensors. In this scheme, an interval can have its value changed either because a direct sensor determines the value or due to the propagation of values through constraints linked of the interval.

This particular feature motivated us to apply `PNF` propagation to the problem of scripting and controling interaction, particularly in the case of immersive, full-body, action-driven systems. The basic idea is to associate actuators and sensors to intervals and to script the interaction by declaring the temporal constraints between those intervals, i.e., whether two intervals must happen in sequence, overlap, or are mutually exclusive. For instance, suppose the arousal of a sensor must trigger an event: this can be described by a *start* ($\{s\}$) temporal constraint between the sensor and the actuator interval. When the triggering event is detected, the `PNF`-state of the sensor interval becomes `now`. By `PNF` propagation on the *start* constraint, the `PNF`-state of the actuator is also set to `now`, provoking the beginning of its associated action.

According to these ideas, an *interval script* describes all the interaction in an application by declaring only the relationships between time intervals corresponding to the actions and to sensor activities. Notice that no explicit time references are needed, for either the duration, start, or finish of an interval.

We see several reasons to use Allen's algebra to describe interaction, especially in the case of story-based and immersive systems. First, no explicit mention of the interval duration or specification of relations between the interval's extremities is required. Second, the existence of Allen's closure algorithm allows the designer to declare only the relevant relations, leading to a cleaner script. Third, the notion of disjunction of interval relationships can be used to declare multiple paths and interactions in an story. In this case, the interval script is in fact the declaration of a graph structure that describes a space of stories and interactions.

Using Allen's temporal structure — including mutually exclusive intervals — constitutes an important advance over current languages for scripting interaction, for example, *Director*, [11], or [4, 6]. However, without the PNF propagation method it would be infeasible to run in real-time any interactive application written with interval scripts. The rest of this section offers a summary of how the run-time management of an application described by an interval script is accomplished (a better description can be found in [17]).

5.1 The Interaction Manager

In the interval script paradigm, the designer has two basic tasks: to define the actual routines corresponding to different *externals* (connectors to real world sensors and actuators) and to determine the relationships between the intervals defined by those externals.

Figure 6 shows the basic structure of the run-time interaction manager. The script defines the relationships between intervals, which are stored in a table, and used by the interaction manager when running the PNF algorithm. The interaction manager considers the PNF-state of all intervals at time $t-1$ to compute the PNF-states at time t. These values are converted and used to call the designer's sensing and actuating routines. For instance, when an interval goes from the fut state to the now state, the designer's routine to start an action or sensing activity is called. The outputs of those routines are mapped back into PNF-states of appropriate intervals, completing the cycle.

When the application is started, the interaction manager sets every interval state to PNF, except for the special interval named start, which is assigned the value N. During run-time, the following basic cycle is repeated till the special end interval becomes N:

1. at the beginning of each cycle, all designer's routines connected to externals are called, considering the PNF-state of each interval to decide which and how each routine should be called;
2. the outputs of the designer's routines, translated into PNF-states, are attributed to the appropriate interval connected to each external;
3. intervals which are not connected to externals (if they exist) have the values in the previous iteration *time-expanded*;
4. the PNF-restriction algorithm is applied, propagating the current values of some intervals through the whole network;
5. if the end interval is not N, the cycle is repeated.

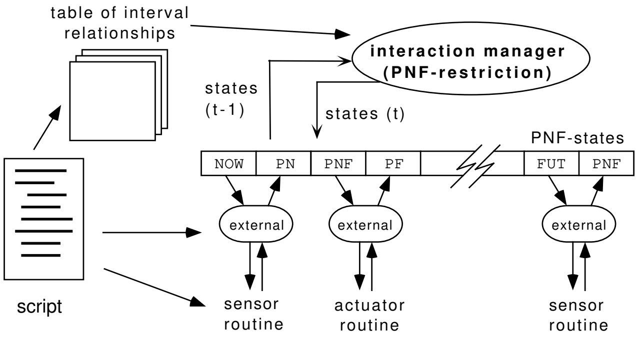

Figure 6: Diagram of the interaction manager.

According to this cycle, if an interval becomes P, it remains with this value until the end of the run. This information is used in the upcoming cycles constraining the values of other intervals and making the system progress through the story defined by the interval script. As in the previous case, before the interaction manager can actually run the interaction described by the interval script, Allen's algorithm must be executed to assure that the relationship between every two intervals is as restricted as possible.

5.2 Two Experiments: *"SingSong"* and *"It/I"*

Interval scripts — and the underlying PNF propagation system — have been used in two interactive, immersive systems called *"SingSong"* and *"It/I"* (built respectively at the ATR Research Laboratories and the MIT Media Laboratory). Both systems integrated human actors and autonomous computer graphics characters in an interactive theatrical play. Figure 7 shows pictures taken during the performances.

"SingSong" is a 4-minute performance where a clown interacts with four screen-projected singers. The interval script used 300 intervals, controling graphics, a MIDI synthesizer, and a camera (see [17] for details). *"It/I"* is a 30-minute performance, followed by audience interaction, where a machine character, played autonomously by the computer system, taunts and plays with a human character. Each of the four scenes of *"It/I"* uses around 150 intervals, handling information from a 3-camera stereo system monitoring the stage, a MIDI synthesizer, a MIDI light-board, and other smaller sensors. *"It/I"* was performed six times for a total audience of about 500 people [16].

6 Final Remarks

This paper introduces a method — PNF propagation — to efficiently exploit the temporal constraints inherent to human action to increase the detection power of simple sensors and to control interaction. The method is based on the specialization into a PNF-network of IA-networks. To detect the occurrence of actions, we consider the sensor values and the past information as unary constraints on the variables of the PNF-network, and calculate an approximation of the minimal domain using an arc-consistency algorithm. In the case of interaction control, intervals are

Figure 7: Images from *"SingSong"*and *"It/I"*.

associated with sensors and actuators which are triggered when those intervals are determined to be happening.

Future work on action recognition includes a more thorough test of the use of PNF propagation by experimenting with a wider range of human actions, and the use of real detectors instead of manually extracted data. We are also developing methods to cope with sensor errors by considering the probability of simultaneous errors and the statistical reliabitility of sensors. A first proposal is discussed in [15].

The interval script paradigm is presently being re-examined and improved. In "It/I", the grammar of the scripting language already contained significant improvments, including the ability to directly express events, cycles, and to easily construct hierarchies. The next step involves targetting designers with less knowledge of programming techniques by simplifying the syntax and the conceptual interface of interval scripts.

Acknowledgements

Kenji Mase from the (ATR Research Laboratories) played an important role in the development of the first version of intervals scripts. The research presented in this paper was partially sponsored by DARPA contract DAAL01-97-K-0103. *"SingSong"* was designed and produced at the ATR Media Integration & Communication Research Laboratories; during that period, Claudio Pinhanez was supported by a Starr Foundation grant from the MIT/Japan Program and by ATR Research Laboratories. *"It/I"* was sponsored by the Digital Life Consortium of the MIT Media Laboratory, and we thank to all members of the crew, in particular Prof. Janet Sonenberg, John Liu, Chris Bentzel, Raquel Coelho, Leslie Bondaryk, Freedom Baird, Richard Marcus, Monica Pinhanez, Nathalie van Bockstaele, and Joshua Pritchard.

References

[1] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[2] James F. Allen and George Ferguson. Actions and events in interval temporal logic. *Journal of Logic and Computation*, 4(5):531–579, 1994.

[3] Matthew Brand, Nuria Oliver, and Alex Pentland. Coupled hidden Markov models for complex action recognition. In *Proc. of CVPR'97*, Puerto Rico, USA, 1997.

[4] M. Cecelia Buchanan and Polle T. Zellweger. Automatic temporal layout mechanisms. In *Proc. of ACM Multimedia'93*, pages 341–350, Ahaheim, California, August 1993.

[5] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1–3):225–233, May 1991.

[6] Rei Hamakawa and Jun Rekimoto. Object composition and playback models for handling multimedia data. In *Proc. of ACM Multimedia'93*, pages 273–281, Ahaheim, California, August 1993.

[7] David Israel, John Perry, and Syun Tutiya. Actions and movements. In *Proc. of the 12th IJCAI*, pages 1060–1065, Sydney, Australia, August 1991.

[8] Vipin Kumar. Algorithms for constraint-satisfaction problems: a survey. *AI Magazine*, 13:32–44, 1992.

[9] Y. Kuniyoshi and H. Inoue. Qualitative recognition of ongoing human action sequences. In *Proc. of IJCAI'93*, pages 1600–1609, 1993.

[10] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[11] MacroMind Inc. *Director's User Manual*. 1990.

[12] Itay Meiri. Combining qualitative and quantitative constraints in temporal reasoning. *Artificial Intelligence*, 87(1–2):343–385, November 1996.

[13] Hans-Hellmut Nagel. A vision of 'vision and language' comprises action: An example from road traffic. *Artificial Intelligence Review*, 8:189–214, 1995.

[14] Claudio S. Pinhanez and Aaron F. Bobick. Approximate world models: Incorporating qualitative and linguistic information into vision systems. In *AAAI'96*, pages 1116–1123, Portland, Oregon, August 1996.

[15] Claudio S. Pinhanez and Aaron F. Bobick. Human action detection using PNF propagation of temporal constraints. Technical Report 423, M.I.T. Media Laboratory Perceptual Computing Section, April 1997.

[16] Claudio S. Pinhanez and Aaron F. Bobick. "It/I": A theater play featuring an autonomous computer graphics character. Technical report # 455, M.I.T. Media Laboratory Perceptual Computing Section, January 1998.

[17] Claudio S. Pinhanez, Kenji Mase, and Aaron F. Bobick. Interval scripts: A design paradigm for story-based interactive systems. In *CHI'97*, pages 287–294, Atlanta, Georgia, March 1997.

[18] Charles J. Rieger III. Conceptual memory and inference. In *Conceptual Information Processing*, chapter 5, pages 157–288. North-Holland, 1975.

[19] Roger C. Schank. Conceptual dependency theory. In *Conceptual Information Processing*, chapter 3, pages 22–82. North-Holland, 1975.

[20] Jeffrey Mark Siskind. *Naive Physics, Event Perception, Lexical Semantics, and Language Acquisition.* PhD thesis, M.I.T, Dept. of Electrical Engineering and Computer Science, January 1992.

[21] Jeffrey Mark Siskind. Grounding language in perception. *Artificial Intelligence Review*, 8:371–391, 1994.

[22] Peter van Beek. Reasoning about qualitative temporal information. *Artificial Intelligence*, 58(1–3):297–326, December 1992.

[23] Marc Vilain and Henry Kaut. Constraint propagation algorithms for temporal reasoning. In *Proc. of AAAI'86*, pages 377–382, Philadelphia, Pennsylvania, 1986.

[24] Marc Vilain, Henry Kautz, and Peter van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. In Daniel S. Weld and Johan de Kleer, editors, *Readings in Qualitative Reasoning About Physical Systems*, pages 373–381. Morgan Kaufmann, San Mateo, California, 1990.