Online Behavior Recognition: A New Grammar Model Linking Measurements and Intents

Nicolas Vidal, Patrick Taillibert, Samir Aknine

Online Behavior Recognition: A New Grammar Model Linking Measurements and Intents

Nicolas Vidal[1,2]
[1]*LIP6*
UPMC
Paris, France
{*name*}.{*surname*}@*lip6.fr*

Patrick Taillibert[2]
[2]*Artificial Intelligence Laboratory*
THALES Airborne Systems
Elancourt, France
{*name*}.{*surname*}@*fr.thalesgroup.com*

Samir Aknine[3]
[3]*LIESP*
UCBL Lyon 1
Lyon, France
{*name*}.{*surname*}@*recherche.univ-lyon1.fr*

***Abstract*—In a maritime area supervision context, we seek providing a human operator with dynamic information on the behaviors of the monitored entities. Linking raw measurements, coming from sensors, with the abstract descriptions of those behaviors is a tough challenge. This problem is usually addressed with a two-stepped treatment: filtering the multidimensional, heterogeneous and imprecise measurements into symbolic events and then using efficient plan recognition techniques on those events. This allows, among other things, the possibility of describing high level symbolic plan steps without being overwhelmed by low level sensor specificities. However, the first step is information destructive and generates additional ambiguity in the recognition process. Furthermore, splitting the behavior recognition task leads to unnecessary computations and makes the building of the plan library tougher.**

Thus, we propose to tackle this problem without dividing the solution into two processes. We present a hierarchical model, inspired by the formal language theory, allowing us to describe behaviors in a continuous way, and build a bridge over the semantic gap between measurements and intents. Thanks to a set of algorithms using this model, we are able, from observations, to deduce the possible future developments of the monitored area while providing the appropriate explanations.

***Keywords*-behavior recognition; activity recognition; constraint programming; pattern recognition; formal grammars**

I. Introduction

Today, numbers of key applications involve the use of wide area maritime surveillance. Those applications can be either civil (sea traffic regulation, exclusive fishing area enforcement, maritime search and rescue missions, illicit trade prevention, ...) or military (embargo situation enforcement, military training area protection, protection of territorial waters, ...).

In order to perform this surveillance, we need information about the monitored entities and about the parameters of the area. The gathering of this data is realized by several means (maritime patrol aircrafts, satellites, frigates, radar stations, ...). This sensor network generates raw heterogeneous imprecise and incomplete measurements. In the end, all this data is transmitted to a human operator, whose task is to determine whether or not actions should be considered. Given the number of parameters and entities involved in a maritime area surveillance task, providing human operators with a comprehensive analysis of the many entities behaviors and intents is mandatory for them to have a real understanding of the complex situation. Furthermore, as we want them to be able to prevent some dangerous state of affairs from happening, we must focus on the online analysis of the situation. Thus, we intend to provide human operators with hypotheses on what behavior type(s) each entity can be performing. In order to do that, we must link the observations provided by a real sensor network with the relevant behavior types.

In the next section we discuss some related works and see why they don't fulfill all the requirements imposed by our application problem. We first introduce some concepts (measurement, view, behavior, ...) in Section 3. Then, we detail our model in Section 4 which is based upon an extension of the attribute formal grammar model [1]. As in [2], the plan library is represented by a grammar depicted in our formalism. A plan is a set of grammar rules. A grammar rule expresses the temporal decomposition of a symbol in terminal and non-terminal symbols. We also clarify some improvements to this model. The confrontation between measurements and a grammar is depicted in Section 5. This is done thanks to two mechanisms: the progressive building of an abstract syntax tree and the handling of attributes relations with a constraint solver. We show how the inherent imprecision on continuous values is managed thanks to an interval constraint solver [3]. We present some of our experiments in Section 6 and we finally conclude in Section 7 with the depiction of ongoing researches.

II. Related Work and Motivation

Techniques used to determine an entity's internal state given external observations are referred to as plan recognition [4]. Most of the time, these observations are represented by a set of symbolic events [5]. These events are used in the depiction of abstract plan steps. A plan is usually a hierarchy of plan steps. In behavior/activity recognition, all the sequences of actions (behaviors) that an entity may be

performing can be generated from the plan library. Recognizing an ongoing behavior, before it has ended, is referred to as dynamic or online plan recognition. A common way to address this problem is to provide at any time hypotheses about the entity behavior (which are revised each time a new observation arrives).

The first challenge in dynamic plan recognition (as in plan recognition), is to design the plan library. Many aspects are relevant when choosing the description model of the plans. For example, temporal relations should be explicitly represented in the model. Usually, the order of plan steps is sufficient, like in Hierarchical Task Networks [6] but some models [7] allow complex temporal relations between events and plan steps if needed. As a plan step can usually be made up of several sub-plan steps, a certain form of hierarchy must be represented in the model, that's why raw Hidden Markov Models [8] encounter some difficulties in recognizing behaviors which require long term interactions. Of course, the more expressive the model is, the less efficient the hypothesis generation mechanism will be [9].

The second difficulty is to manage ambiguous cases generating a high number of recognition hypotheses. This problem has previously been addressed. In most approaches, a probabilistic extension of an existing model is proposed like [10] or [11] with Stochastic Grammars. The probabilistic solution allows the pruning of low likelihood hypothesis and the ranking of the remaining ones. Classical probabilistic approaches used in pattern recognition like Hidden Markov Models [8] have been used to capture the hierarchical essence of plans, extensions like Hierarchical Hidden Markov Models [12] or Abstract Hidden Markov Memory Models [13] have been proposed. All those probabilistic models can be compared on a common ground thanks to Dynamic Bayesian Networks [14]. But this likelihood ranking is not a shared feature. Indeed, for some applications relating to abnormal behavior detection, sometimes, a low-probability behavior should be highlighted instead of being pruned. Thus, [15] introduced the concept of capabilities-based plan recognition, bringing a new concept (capabilities) in a classical Bayesian plan recognition context. Recently, [16] focused on allowing the plan library builder to add some decisional criteria on plans and plan steps. The idea is that the relevance of a hypothesis is linked to the observer and that it should be specified directly into the plan library.

All those previous methods have focused on handling the effects of ambiguous cases. None are wondering why those ambiguous situations are happening. What if a wise choice of representation model was helpful in avoiding the generation of a great number of hypotheses?

First, let's remind that plan recognition models are often based on symbolic events. When dealing with real world applications, observations are not a set of symbolic events. The captured data coming from a sensor network is heterogeneous, multidimensional, can be incomplete and if we are dealing with numerical measurements, imprecise. With regards to that, using the above legacy plan recognition techniques requires to have a mechanism, transforming those observations into events. When addressing this issue, the plan recognition process is assumed to be divided in two steps: first the events generation and then the confrontation of events against the plan library. For instance, in [2] a tracking system extracts some visual atomic events from a video surveillance camera stream (like the appearance of objects) and then feeds a stochastic attribute grammar parsing algorithm with them. In this work, no comments were made about the cost of computing those events by the tracking system. In fact, very few people acknowledge the event generation process as a part of plan recognition.

[17] focused on the best way to compute, from these raw observations, symbolic events in order to use legacy symbolic plan recognition techniques. They build the event generator (in this case a feature decision tree) function of the plan library before the recognition process begins. We consider that this work is fundamental, it exhibits the existence of a strong relation between the event generation process dealing with raw measurements and the legacy plan recognition process. The fact is that with tightening up the links between the two phases, an overall benefit has been drawn out. However, this decomposition still generates several issues.

(1) First, during the recognition process, the link between the two processes is still unidirectional: the event generator computes all the events it can produce from the observations and gives them to the legacy plan recognition process. Some of those events don't need to be computed. Assuming the only one recognition hypothesis remaining about the behavior of a boat is that it is going toward Algiers, generating an event concerning the weather in New York is very unlikely to be relevant. It would be better to generate only the relevant events function of the plan recognition state.

(2) Secondly, the first process causes an information loss which engenders ambiguity in the plan recognition step. The application context we are in is already deficient in workable data. For example, reducing the real value of the measured orientation ($-79°$) of a boat in a symbolic one (west) is information destructive and this can lead to producing several hypotheses (the boat is going toward harbor A, B or C). In order to deal with the resulting huge number of recognition hypothesis, some have developed ranking or pruning techniques based upon likelihood [18] or decisional [16] criteria. Still, it would have been better to keep the real measured value.

(3) Lastly, the limit between real observations and symbolic events is the same for all the behavior types represented in the plan library. Though, all those don't involve the same concepts and relations, and if it was not for the sake of other behavior types, some would have been described differently with a more appropriate separation between the real data

and the abstract layers. For example, let's assume we want to describe two behaviors for a boat: a sudden acceleration and a supply delivery. The first behavior should be described with speed change events ($speed\_augmentation$) whereas the second behavior should be described with events of higher abstraction ($sailing\_toward\_a\_harbor$, $loading\_supply$, ...). Indeed, when dealing with a huge number of behavior types, finding a suitable common event ground for the description of the plan library involves too much compromises between efficiency and declarativity.

We argue that the source of those problems is the event generation process (multidimensional numeric-symbolic measurements to unidimensional symbolic events). Introducing a synthetic abstraction level introduces more issues than it solves. When applied to real world applications like ours, we can't afford to realize unnecessary computations, to generate more than the inherent knowledge domain ambiguity, and to add some difficulties to the plan library building process.

[19] focused on how an activity recognition process could be working directly with low level heterogeneous events and proposes a homogeneous model to represent a hierarchy of facts (which could be associated to a plan library). However, they didn't go far enough. In their formalism, some relations can be described between numeric and symbolic data. But even if numeric events are taken into account, they don't handle imprecise or missing measurements. In fact, whereas their event-level is closer to the sensor measurements, there is still a two-stepped treatment in their approach.

We choose to avoid the introduction of the classical event level, splitting the measurements handling and the plan recognition process. While doing so, we are bound to directly use raw measurements in the second step. As a result, we must be able to handle heterogeneous multidimensional and possibly imprecise data and describe in the same model our abstract plan steps.

Let's consider a motivating example. It consists in a scenario of a sailing boat tacking from a harbor (Bastia) toward a second one (Livorno). These kinds of behaviors are complex: they involve multiple steps (leaving Bastia through its fairway, tacking toward Livorno's fairway, and entering Livorno), external parameters like wind orientation must be taken into account, and a combination of multiple geometrical relations is used. In this situation, the only live information at our disposal are imprecise (and maybe incomplete) measurements about the sailing boat position and orientation issued from a sensor network (radars, spy planes, etc.) from time to time.

To achieve our goal, we present a new generative model capable of representing in a declarative and continuous way this kind of behavior. Instances of this model are then confronted to raw measurements to determine whether or not the monitored entity is performing the depicted behaviors, without generating more recognition hypotheses than needed. Furthermore, thanks to this declarative model we are able to provide explanations justifying them and predictions on the future evolution of the entity's behavior. Confronted to this triple result, a human operator is able to evaluate the local relevance of a hypothesis, to anticipate and prevent unwanted situations.

III. PRELIMINARY CONCEPTS

In real world applications, live data is provided by a heterogeneous sensor network. In order to be able to deal in a consistent way with heterogeneous data, we first define a common abstraction of the concept of measurement:

Definition 1: A measurement is defined as a 4-tuples $\langle E, P, S, V \rangle$, with E the entity unique identifier, P the name of the measured property, S the sensor unique identifier, and V the measurement value. This value can be either a real numeric value or a symbol.

Measurements can't be carried out at all times, and for an arbitrary time T, we can't be sure to have a value for all the entities properties measured by our sensor network. Thus, we suppose the existence of an interpolation/extrapolation system, which will generate from previous measurements some complete views of the entities properties at a given time T.

Definition 2: A view is defined as the set M of all the measurements done or extrapolated at a given time.

Definition 3: A subview is defined as a subset of a view.

In order to represent the imprecision induced by the sensors and the interpolation/extrapolation system, we allow the numeric value of a measurement to be an enclosing interval of real numbers. In the case of uncertainty about a symbolic value we allow it to be a set of symbols. Of course, if a measurement is missing, and can't be inter/extrapolated, it is replaced by either $]-\infty, +\infty[$ in case of a numeric value, or S in case of a symbolic one, such as S is the set of symbols. Here is a sample of a subview made up of two measurements (speed and type of the boat named Artemis II):

$\langle (ArtemisII, Speed, Radar01, [15.28, 17.67]),$
$(ArtemisII, Type, Radar02, \{CargoShip, Sailboat\}) \rangle$

We then define a behavior for an entity as a sequence of subviews, where there is at least one measurement about one of its properties.

Definition 4: A behavior for an entity e is defined as a sequence of subviews $(M_1, \ldots, M_n)$ where there is at least one $m \in M_1 \cup \ldots \cup M_n$ such as $m = \langle e, P, S, V \rangle$.

In classical online plan recognition, a plan is a sequence of events. Having at our disposal a plan library, the goal is to find which plans match with an incoming sequence of events. In our case, the sequence of events is a sequence of views and all the plans (i.e. sequence of subviews) can't be exhaustively written in the plan library.

The previous concepts are intended to be very general and could be used through a variety of application domains. With such concepts, it is obvious that we are not going to be able to list exhaustively all the behaviors that we want to recognize. Furthermore, recognizing one behavior in particular is never useful at such a low level of description. However recognizing all the behaviors with some common characteristics (ex: all the behaviors in which the boat is sailing toward a precise harbor) would be. Those characteristics are often abstract ones and are never easily linked with low level measurements.

The first problem is to find a representation in which we can describe all the characteristics, function of those measurements. Then, with such a representation, how can we be able to match an instance of it with live data? We address these two issues in the next sections.

IV. MODEL

Any plan recognition system works with a collection of plans to be recognized. These plans must be described through a formal language. In this section we define the concepts behind our plan library model.

Our model is an extension of the attribute formal grammar model.

Definition 5: An attribute grammar is defined as a 5-tuples $\langle N, T, R, S, A\rangle$ with N a set of non-terminal symbols, T a set of terminal symbols, $R = \bigcup_{i\in N} R^i$ where R^i is the set of production rules associated with each non-terminal symbol $i \in N$, S a set of starting symbols with $S \subseteq N$ and $A = \bigcup_{j\in N\cup T} A^j$ where A^j is the set of attributes associated with each symbol $j \in N\cup T$ [1].

In a classical attribute grammar, a production rule $r \in R^{s_0}$ is divided in a left part made up of a non-terminal symbol $s_0 \in N$ with its attributes A^{s_0}, a middle part made up of a sequence of terminal and non-terminal symbols $s_1, s_2, s_3, \ldots, s_n \in N \cup T$ associated with their respective attributes and a right part made up of equality relations between the symbols attributes involved in the rule. The syntactic symbols used to split those parts are $\rightarrow$ and :.

Example 1: In the classical attribute grammar model, a production rule $r \in R^{s_0}$ is written :
$s_0(a_1^{s_0}, a_2^{s_0}, a_3^{s_0}, a_4^{s_0}) \rightarrow s_1(a_1^{s_1}, a_2^{s_1}) \ldots s_n(a_1^{s_n})$
$: a_1^{s_0} = a_2^{s_2}, \ldots, a_3^{s_0} = a_1^{s_n}$

[1]As usual, a terminal symbol is written in lower case characters and a non terminal symbol begins with an upper case character.

In the following, we extend the attribute grammar model with new definitions for R, T and A, and introduce the concept of Terminal Set.

A. The & symbol

As we already stated, our behavior library will be encoded through a formal grammar. We need a way to combine several sub behaviors such as $Acceleration$ and $SailingTowardAnEntity$ to compose a new one such as $DeliberateCollisionCourse$. Thus we introduce some changes in the model of R.

Definition 6: A language $\mathbb{L}^G$ is defined as the set of sequences of terminal symbols that can be generated/recognized by a formal grammar G.

A production rule represents a decomposition of a non terminal symbol into other symbols (non terminal and terminal ones). To obtain a sequence of terminal symbols belonging to the grammar language, beginning with a start symbol, each non terminal symbol is replaced recursively according to one of its production rules until there only subsists terminal symbols.

Definition 7: A sub-language $\mathbb{L}_n^G$ is defined as the set of sequences of terminal symbols that can be generated/recognized by a non terminal symbol $n \in N$ of a given formal grammar G.

We are going to use two symbols in the middle part of our rules. The first one is $\triangleleft$ expressing the classical sequentiality of symbols and the second one is the intersection symbol &.

Definition 8: Given a formal attribute grammar $G = \langle N, T, R, S, A\rangle$ and $n_1, n_2 \in N$, the sub-language $\mathbb{L}_{n_3}^G$ generated/recognized by $n_1 \& n_2$ is such as $\mathbb{L}_{n_3}^G = \mathbb{L}_{n_1}^G \cap \mathbb{L}_{n_2}^G$

Example 2: Let $G = \langle\{I, A, B\}, \{a, b\}, \{I \rightarrow A\&B :), (A \rightarrow a \triangleleft a \triangleleft a :), (B \rightarrow b \triangleleft b :), (B \rightarrow (a \triangleleft a \triangleleft a :)\}, \{I\}, \{\}\rangle$ a formal grammar, then $\mathbb{L}^G = \{(a \triangleleft a \triangleleft a)\}$.

Remark 1: Using this symbol we can easily merge two grammars $G1 = \langle N1, T1, R1, S1, A1\rangle$ and $G2 = \langle N2, T2, R2, S2, A2\rangle$ in a new one $G = \langle N, T, R, S, A\rangle$ such as $N = N1 \cup N2 \cup \{Snew\}$, $T = T1 \cup T2$, $R = R1 \cup R2 \cup \bigcup_{s1\in S1, s2\in S2}\{Snew \rightarrow s1\&s2\}$, $S = \{Snew\}$ and $A = A1 \cup A2$.

Merging two grammars can be useful in order to verify several simultaneous properties at once (ex. : respects of language syntax and code standards at once).

B. Terminal Sets

As previously stated, we want to analyze a sequence of $views$ and not a sequence of terminal symbols. Thus, in

our grammar model, the elements of T are not terminal symbols, but $terminal\ sets$ of elementary symbols.

Definition 9: An attribute formal grammar G depicted in our model is a 6-tuples such as $G = \langle N, \Theta, T, R, S, A\rangle$, with N a set of non terminal symbols, Θ a set of elementary symbols, T a set of terminal sets such as $T \subseteq P(\Theta)$, $R = \bigcup_{i\in N} R^i$ where R^i is the set of production rules associated with each non-terminal symbol $i \in N$, S a set of starting symbols with $S \subseteq N$ and $A = \bigcup_{j\in N\cup\Theta} A^j$ where A^j is the set of attributes associated with each symbol $j \in N \cup \Theta$.

Example 3: Let G=$\langle\{I, A, B\}, \{a, b, c\}, \{\begin{pmatrix}a\\b\\c\end{pmatrix}, (a), \begin{pmatrix}b\\c\end{pmatrix}\}$, $\{(I\rightarrow A\&B :),\ (A\rightarrow \begin{pmatrix}a\\b\\c\end{pmatrix} \triangleleft \begin{pmatrix}a\\c\\b\end{pmatrix} \triangleleft \begin{pmatrix}b\\c\\a\end{pmatrix} :),\ (B\rightarrow \begin{pmatrix}b\\c\end{pmatrix} \triangleleft (a) :),$ $(B\rightarrow \begin{pmatrix}a\\b\\c\end{pmatrix} \triangleleft \begin{pmatrix}a\\b\\c\end{pmatrix} \triangleleft \begin{pmatrix}a\\b\\c\end{pmatrix} :)\},\ \{I\},\ \{\}\rangle$ a formal grammar, then $\mathbb{L}^G = \{(\begin{pmatrix}a\\b\\c\end{pmatrix} \triangleleft \begin{pmatrix}a\\b\\c\end{pmatrix} \triangleleft \begin{pmatrix}a\\b\\c\end{pmatrix})\}$.

A terminal set represents a view. It's near as impossible to specify all the combinations of measurements a view can contain. We introduce an operator [], $[\theta]$ stands for the belonging of a symbol $\theta \in \Theta$ to a terminal set $t \in T$.

Definition 10: Given a formal attribute grammar $G = \langle N, \Theta, T, R, S, A\rangle$, $\mathbb{L}^G_{[\theta]} = \{(t)|t \in T \wedge \theta \in \Theta \wedge \theta \in t\}$.

We introduce a particular usage of this operator with the symbol $*$, which stands for any symbol.

Definition 11: Given a formal attribute grammar $G = \langle N, \Theta, T, R, S, A\rangle$, $\mathbb{L}^G_{[*]} = \bigcup_{t\in T}\{(t)\}$.

Example 4: Let $G = \langle\{I\}, \{a, b, c\}\ \{\begin{pmatrix}a\\b\\c\end{pmatrix}, (a), \begin{pmatrix}b\\c\end{pmatrix}\}$, $\{(I \rightarrow [b] \triangleleft [c] :),\ \{I\},\ \{\}\rangle$ a formal grammar, then $\mathbb{L}^G = \{(\begin{pmatrix}a\\b\\c\end{pmatrix} \triangleleft \begin{pmatrix}a\\b\\c\end{pmatrix}), (\begin{pmatrix}b\\c\end{pmatrix} \triangleleft \begin{pmatrix}a\\b\\c\end{pmatrix}), (\begin{pmatrix}b\\c\end{pmatrix} \triangleleft \begin{pmatrix}b\\c\end{pmatrix}), (\begin{pmatrix}a\\b\\c\end{pmatrix} \triangleleft \begin{pmatrix}b\\c\end{pmatrix})\}$.

C. Attributes

In our model, an attribute is a name/value pair. Tts name is a string of characters, and its value can be one of two types, symbolic or numerical (a real number).

Notation 1: If A is a set of attributes, let $values(A)$ be the set of values and $names(A)$ be the set of names of all attributes of A.

Definition 12: An attribute set A^i for a symbol $i \in N\cup\Theta$ is such as $A^i = A^i_s \cup A^i_n$ where $values(A^i_n) \subset \mathbb{R}$ and $values(A^i_s)$ is set of strings of characters.

Notation 2: A terminal set $t \in T$ containing a elementary symbol $a \in \Theta$ with at least two attributes named $n1$ and $n2$ with respectively values $v1$ and $v2$, is noted $[a(n1 : v1, n2 : v2)]$.

D. Attributes Relations

Notation 3: Let $r \in R$ a production rule of a given grammar G, the set of non-terminal symbols used in the left and middle part of r is noted N^r, the set of elementary symbols belonging to terminal sets involved in the middle part of r is noted Θ^r and the set of relations between attributes in the right part of r is noted C^r.

Relations between attributes are described in the right part of a production rule. As there are two kinds of attributes, there are two kinds of attribute relations. Relations between symbolic attributes and between numerical ones.

Definition 13: Let $r \in R$ a production rule of a given grammar G. C^r is such as $C^r = C^r_s \cup C^r_n$ where C^r_s is a set of equality relations between the elements of $\bigcup_{i\in N^r\cup\Theta^r} A^i_s$ and C^r_n is a set of arithmetical relations between the elements of $\bigcup_{i\in N^r\cup\Theta^r} A^i_n$.

Usually, attributes are splitted into two groups, inherited and synthesized attributes and don't influence the syntactic parsing step. We don't make this distinction in our model as an attribute can be used in both situations in the very same grammar. Furthermore, attribute relations, are not just functions used to evaluate some semantics values associated with the rules. They must be considered as conditions for the production rule to be enforced. Therefore, during the generation/recognition phase, a production rule can be used to reduce a non-terminal symbol only if all its relations between attributes are not proven wrong (least commitment hypothesis).

E. Production Rule Samples

Formal grammars are used through a variety of application domains, mostly in order to recognize given patterns over a spatial structure, like a text file. We are going to use our model in order to describe and recognize temporal patterns (behavior types).

The instanciation of the previous abstract concepts with our application is the following :

- Terminal Sets $\Leftrightarrow$ Subviews.
- Symbols of Terminal Sets $\Leftrightarrow$ Measurements.
- A Formal Grammar $G \Leftrightarrow$ The description of a behavior type.
- $\mathbb{L}^G \Leftrightarrow$ All the sequences of subviews belonging to a behavior type.

Here is some samples of production rules pointing out the relevance of the extensions we made.

```
GoToHarbor(Boat: B,Harbor1: H1,
           Harbor2: H2, StartTime: ST,
           EndTime: ET,MaxTime: MT)→
 CurrentTime(Time: ST)◁
 LeaveHarbor(Boat: B,Harbor: H1)◁
 SailTowardHarbor(Boat: B,Harbor: H2)◁
 EnterHarbor(Boat: B,Harbor: H2)◁
 CurrentTime(Time: ET).
 : MT<ET-ST.
```

In the above example, we are describing a rule representing a possible high level behavior for an entity *Entity* going from a first harbor *Harbor1* toward a second one *Harbor2* in less than a predefine amount of time *MaxTime*.

Views are filled with only one symbol : $measurement$. Of course, it would be tough to build a useful grammar with just this symbol at our disposal. That's why attributes corresponding to the fields of the previous definition of a measurement ($Entity$, $Property$, $Sensor$, $Value$) are associated with this terminal symbol. Thus, if we want to use the speed of an entity we just have to write:

```
Speed(Entity: E,Value: V,Property: P)→
 [measurement(Entity: E,Property: P,
              Value : V)].
 : P="speed".
```

The introduction of our intersection symbol $\&$ lets us write relations between a set of attributes representing entities properties at a given time (multi-dimensionality) such as:

```
AtHarbor(Boat: B,Harbor: H,
         BoatType: BT,HType: HT,
         BoatPosX: Bx,BoatPosY: By,
         HarborPosX: Hx,HarborPosY:Hy)→
 Position(X: Bx,Y: By,Entity: B)&
 Position(X: Hx,Y: Hy,Entity: H)&
 Type(Entity: B,Type: BT)&
 Type(Entity: H,Type: HT).
 : BT="SailingBoat",HT="Harbor",
   Bx=Hx,By=Hy.
```

This rule describes the fact for a sailing boat $Boat$ to be at a harbor $Harbor$. As we already stated, our behavior types often involve classical geometrical relations. Trying to describe that a boat is heading toward a point implies being able to represent vectors and manipulate them. Here is a well known geometrical relation in 2-D space:

```
FrameOfRefSwitch(OriginX: Ox,OriginY:Oy,
                 X: X,Y: Y,
                 NewX: NX,NewY: NY,
                 OriginAngleX: OAx,
                 OriginAngleY: OAy)→
 [*].
 : NX=(X-Ox)*OAx+(Y-Oy)*OAy,
   NY=(Y-Oy)*OAx-(X-Ox)*OAy,
   X=NX*OAx-NY*OAy+Ox,
   Y=NX*OAy+NY*OAx+Oy.
```

This is a kind of relations, useful in our application context, which can't be specified in most of current plan recognition models. Hopefully, those relations only need to be written once, and can be propagated in all the grammars they are needed in. Here is an example of usage of the above-written geometrical relation:

```
InRectangleArea(AreaCenterX: Ax,
                AreaCenterY: Ay,
                PointPosX: Px,
                PointPosY: Py,
                PointAreaPosX: PAx,
                PointAreaPosY: PAy,
                AreaWidth: AW,
                AreaLength: AL,
                AreaAngleX: AAx,
                AreaAngleY: AAy)→
 FrameOfRefSwitch(OriginX: Ax,
                  OriginY: Ay,
                  X: Px,Y: Py,
                  NewX: PAx,NewY: PAy,
                  OriginAngleX: AAx,
                  OriginAngleY: AAy).
 : abs(PAy)<abs(AW)/2 ,
   abs(PAx)<abs(AL)/2
```

This rule describes the fact for a point ($PointPosX$, $PointPosY$) of being inside a rectangular area defined by its center ($AreaCenterX$, $AreaCenterY$), its width ($AreaWidth$), its lenght ($AreaLength$) and oriented on an axis of direction ($AreaAngleX$, $AreaAngleY$).

V. MODEL USAGE

After having described all the relevant behaviors in our model, we now have at our disposal a complete grammar, standing for our plan library. We are now going to explain how, from this grammar and the succession of $views$ provided by our interpolation/extrapolation system, we can deduce hypotheses about the entity behavior(s). The algorithm will answer this question: Is our grammar capable of generating a behavior (a succession of subviews involving at least one measurement related to the entity) that matches the observed behavior ?

Of course answering this question isn't enough for our application goal which is to provide human operators with a comprehensive analysis of the many entities behaviors and intents. Therefore, the result of our algorithm must be triple: $What$? (behaviors the entity can be performing), How? (the hypotheses under which the system is giving this result) and $NextSteps$? (the future possible evolutions of the monitored area).

Providing an answer to the question *Could a sentence have been produced by a given grammar ?* is referred as parsing. Usually in the Formal Language Theory, during the parsing step, a derivation tree is built. We are going to follow this process and build the derivation trees of our grammar that can produce the observed sequence of views. As we are dealing with online plan recognition, the algorithm we propose is iterative and produces derivation trees as temporary results.

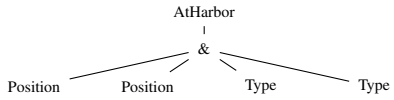
A. Initialization

We initialize L, the list of derivation trees (hypotheses), as a singleton containing the tree made up of one node S, standing for the start symbol of our grammar (in our case it will be the non-terminal $Behavior$).

B. Iteration

As shown in Figure 1, a grammar rule can be viewed as a subtree: the root is the left symbol, children are the symbols in the right part. In the case of symbols linked simultaneously in the right part, an intermediate $\&$ node is inserted.

Figure 1. Conversion of the $AtHarbor$ rule into a subtree



Then, each time a view V is produced by our interpolation/extrapolation system, one iteration of the following algorithm is applied:

```
1: Each tree of L is partially developed
  -A node is expanded according to its
   grammar rule.
  -If there are several rules for a
   node, the whole tree is reused as
   many times as needed.
  -The left child of non-terminal nodes
   is expanded recursively.
  -All the children of & node are
   expanded recursively.

2: Matching with views:
  -Associate each leaf of each tree
   (terminal symbol) with a measurement
   of the current view.
  -If several combinations are possible,
   reuse the tree as many times as
   needed.

3: Repositionning:
  -Locate the next node(s) to be
   expanded for the next iteration.
```

Figure 2. Partial development of a tree after the first step of the iteration

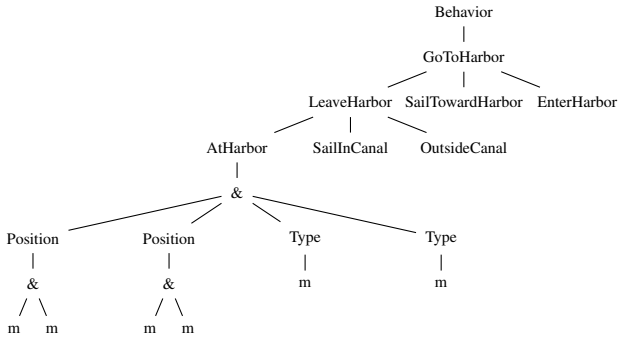
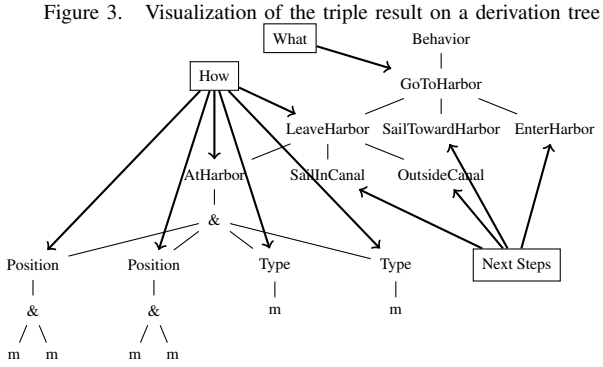


Figure 2 shows the result of the first step of the iteration (m stands for $measurement$). It is noticeable that the tree isn't fully deployed and that all the expanded nodes in this iteration are involving $measurements$ belonging to the same $view$.

Through the analysis of the derivation tree, we can extract all the information we need for our triple result (this is illustrated on the derivation tree on Figure 3): $What$ as the direct child of root $Behavior$, How as all the expanded nodes of the tree and the $NextSteps$ as all the nodes not yet expanded.

Figure 3. Visualization of the triple result on a derivation tree



C. Attribute relations management

Unlike [2] we provide mechanisms for imprecision and uncertainty handling, which are essentials in order to deal with raw measurements. As we previously said, we use constraint programming techniques in order to deal with the relations linking the attributes of our nodes. In order to do that, we associate a constraint store to each derivation tree. When a tree is reused, the associated constraint store is reused too. Each time a node is expanded, the relations between attributes associated with the rule involved are inserted into the constraint store. If a constraint store is

inconsistent, the associated derivation tree is removed from the tree list (the hypothesis is invalid).

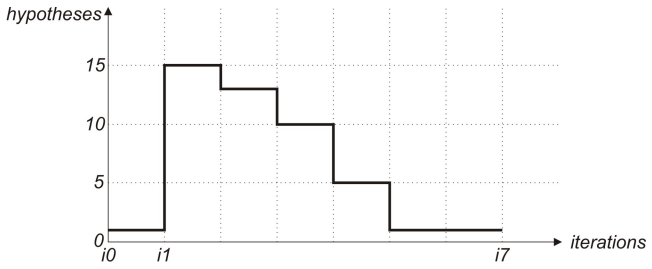
As we said earlier, the observed data can be imprecise, and numeric measurement values are represented as intervals or real numbers. Thus, we use an Interval Constraint Solver to manage this kind of data. Hence non-linear mathematical relations are allowed.

This way, it is not necessary to handle imprecision in the description of the plan library (the grammar).

VI. EXPERIMENTS

We use a simulator allowing us to play predefined scenarii while simulating a sensor network generating imprecise measurements about our scene entities. In order to experiment on a large number of different situations, we chose to confront our method on a leisure sailing scenario. Going from one point toward another one with a sailing boat isn't as easy as it looks. In fact, function of the wind, the boat may need to tack several times, switching from one direction to another one in order to reach a given harbor. The sensor network simulated is a simulation of the Automatic Identification System [20] providing position, heading and speed of boats. Those three simulated imprecise measurements are at our disposal for each entity, thanks to our interpolation/extrapolation system at approximately each hour. We consider intervals of real numbers to be our raw data. Harbor and fairways are also considered as measurements, even if their position will remain unchanged through the time. In the scenario represented on Figure 4, a boat is going from Bastia to Livorno. Fourteen other harbors are considered in order to confront our system with ambiguous cases. The wind speed and orientation, which will remain homogeneous on the monitored area for simplification purposes, is also considered as a measurement. The position of the sailing boat each time a view of the situation is taken is also represented. The $GotoHarbor$ behavior for a sailing boat requires forty rules in order to be characterized with enough precision. We presented some samples of this grammar in the previous sections. Most of the rules are geometrical relations. More domain related rules like ClosedHauled (sailing as near as possible close to the wind) are easily described as function of those relations. Figure 5 shows the evolution of the number of hypotheses (derivation trees) function of the number of iteration of the algorithm (each time a $view$ is received, an iteration is performed). Before $i1$, the boat is in Bastia's fairway and the target harbor is yet unknown. At $i1$ the boat is just leaving the harbor fairway and could be sailing toward each harbor in the area. From $i1$ to $i7$, the boat begins to tack toward Livorno's fairway, hypotheses with other harbors as targets are successively eliminated. In all our experiments, the maximum number of derivation trees after an iteration was linear function of the number of behavior types. However, we must precise that it mostly depends on how the plan library (the grammar) is written. In fact, it is possible to write some artificial grammars in our model that can easily produce an exponential number of hypotheses function of the number of iterations. However, when defining real behaviors into our model, the mechanisms involved in this prohibitive tree generation didn't appear. We found that describing behavior types with great precision was the best way to fight against ambiguity, and a huge number of hypotheses.

Figure 5. Evolution of the number of derivation trees function of the number of iterations (number of $views$ received) during the recognition.



VII. CONCLUSION

In this article, we showed that splitting the behavior recognition process (event generation task and then plan recognition task) engenders several issues like unnecessary computations, generation of more than the inherent knowledge domain ambiguity and difficulties in the plan library building process.

Thus, we have presented a one-stepped approach capable of solving the issues of the behavior recognition problem with real sensor data as live input (multidimensional imprecise heterogeneous measurements).

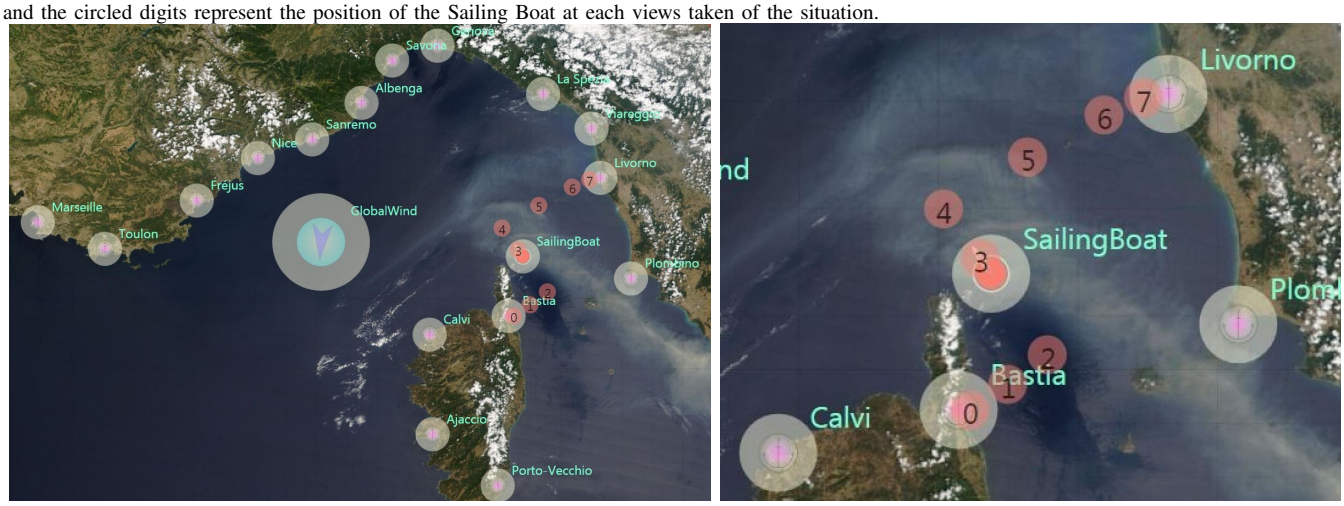
To achieve that, we defined a specific attribute formal grammar model, and used it to build a grammar representing our plan library. Lastly, we showed how such a grammar could be confronted to raw measurements in order to produce a triple result: $What$? (What behaviors the entity can be performing), How? (Under which hypotheses the system is giving this result) and $NextSteps$? (the future possible evolutions of the monitored area).

Even if it wasn't fully specified in this paper, the use of interval constraint programming techniques allows to deal with imprecise measurements without impacting the plan library building process. Furthermore, it is a great tool to handle missing measurements and increases the robustness of our approach.

Our approach is currently implemented and conclusive tests have been carried out. Several grammars made up of about forty rules each are available. The whole has been merged in a demonstrator.

Ongoing researches are numerous. We are going to focus on techniques allowing us to merge several almost identical

Figure 4. Screenshot of the simulation : a sailing boat going from Bastia to Livorno. Harbors are in white, the global wind is blowing South South West and the circled digits represent the position of the Sailing Boat at each views taken of the situation.



hypotheses and on a possible probabilistic extension of our model.

Besides, recognizing interleaved behaviors has been a key feature in some recent behavior recognition system [17]. However, in order to do that, it is usually required to fully deploy before runtime the generative models they are represented in. This is still impossible in our case due to numerous recursive behaviors. We intend in future work to deal with these issues.

REFERENCES

[1] D. Knuth, "Semantics of context-free languages," *Theory of Computing Systems*, vol. 2, no. 2, pp. 127–145, 1968.

[2] S. Joo and R. Chellappa, "Recognition of multi-object events using attribute grammars," in *ICIP*, 2006, pp. 2897–2900.

[3] W. Older and A. Vellino, "Constraint arithmetic on real intervals," in *Constraint Logic Programming: Selected Research*, 1993, pp. 175–195.

[4] C. Schmidt, N. Sridharan, and J. Goodson, "The plan recognition problem," *Artificial Intelligence*, vol. 11, pp. 45–83, 1978.

[5] H. Kautz, "A formal theory of plan recognition," Ph.D. dissertation, University of Rochester, 1987.

[6] S. Adams and A. Goel, "A stab at making sense of vast data," in *PAIR*, 2007, pp. 1–8.

[7] C. Dousson and M. Ghallab, "Suivi et reconnaissance de chroniques," *Revue dintelligence artificielle*, vol. 8, no. 1, pp. 29–61, 1994.

[8] L. Rabiner and B. Juang, "An introduction to hidden markov models," *ieee assp magazine*, vol. 3, no. 1 Part 1, pp. 4–16, 1986.

[9] N. Chomsky, *Aspects of the Theory of Syntax*. MIT press, 1965.

[10] A. Bobic and Y. Ivanov, "Application of stochastic grammars to understanding action," Ph.D. dissertation, 1998.

[11] D. Pynadath and M. Wellman, "Probabilistic state-dependent grammars for plan recognition," in *UAI*, 2000, pp. 507–514.

[12] N. Nguyen, D. Phung, S. Venkatesh, and H. Bui, "Learning and detecting activities from movement trajectories using the hierarchical hidden markov model," in *CVPR*, vol. 2, 2005, pp. 955–960.

[13] H. Bui, "A general model for online probabilistic plan recognition," in *IJCAI*, vol. 18, 2003, pp. 1309–1318.

[14] Z. Ghahramani, "Learning dynamic bayesian networks," *LNCS*, vol. 1387, pp. 168–197, 1998.

[15] R. Suzic and P. Svenson, "Capabilities-based plan recognition," in *ICIF*, Florence, 2006.

[16] D. Avrahami-Zilberbrand and G. Kaminka, "Incorporating observer biases in keyhole plan recognition (efficiently!)," in *AAAI*, 2007, pp. 944–949.

[17] D. Avrahami Zilberbrand and G. Kaminka, "Fast and complete symbolic plan recognition," in *IJCAI*, 2005, pp. 12–20.

[18] C. Geib and R. Goldman, "A probabilistic plan recognition algorithm based on plan tree grammars," *Artificial Intelligence*, vol. 173, no. 11, pp. 1101–1132, 2009.

[19] N. Rota and M. Thonnat, "Activity recognition from video sequences using declarative models," in *ECAI*, 2000, pp. 673–680.

[20] A. Harati-Mokhtari, A. Wall, P. Brooks, and J. Wang, "Automatic identification system (ais): data reliability and human error implications," *The Journal of Navigation*, vol. 60, no. 03, pp. 373–389, 2007.