International Journal on Artificial Intelligence Tools
(1997)
© World Scientific Publishing Company

A PURELY SYMBOLIC MODEL FOR DYNAMIC SCENE INTERPRETATION

Laurent CHAUDRON, Corine COSSART, Nicolas MAILLE, Catherine TESSIER
ONERA-CERT
BP 4025 - 31055 Toulouse Cedex 04 - France

Received 1 May 1997
Revised 8 July 1997

The symbolic level of a dynamic scene interpretation system is presented. This symbolic level is based on plan prototypes represented by Petri nets whose interpretation is expressed thanks to 1st order cubes, and on a reasoning aiming at instantiating the plan prototypes with objects delivered by the numerical processing of sensor data. A purely symbolic meta-structure, grounded on the lattice theory, is then proposed to deal with the symbolic uncertainty issues. Examples on real world data are given.

Keywords: knowledge representation, 1st order logic, Petri nets, scene interpretation, symbolic uncertainty, lattices

1. Introduction

Dynamic scene interpretation from images for surveillance and intelligence systems, autonomous vehicles, decision aid systems, involves symbolic model designing for representing activities and plans that are likely to be observed, so as to give a meaning to the outputs of the numerical image processing.
The aim of PERCEPTION project [1] [1]is to study and develop numerical and symbolic methods and processings allowing a semantically rich interpretation of sensor data to be computed and updated, considering dynamic environments. The perception function ranges from the outputs of the sensors (e.g. black and white, color and infrared cameras, or even "human sensors") to a symbolic representation of the observed environment (say: the current situation), with a feedback considering the management of the perception resources.

This closed loop involves four different modules (Fig. 1):

- the Numerical Level (NL) aims at delivering labelled and tracked objects from sensor data. The detection of moving objects is performed thanks to the cooperation of static and dynamic segmentations of images; the labelling results from the fusion of different classifiers and takes advantage of the sensor complementarity.

[1]See also http: //www.cert.fr/francais/dera/dera/PERCEPTION/perception.html

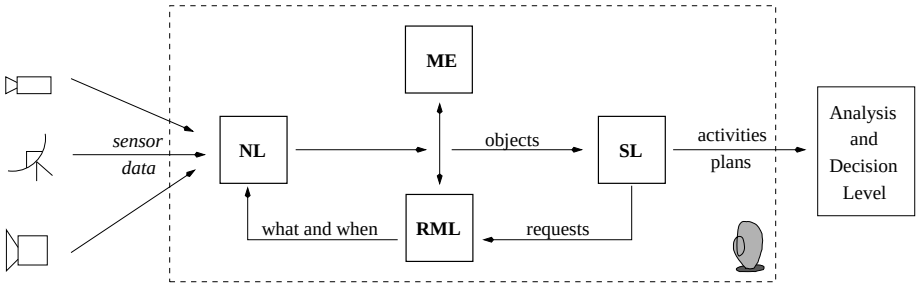

Fig. 1. Overview of the perception function (PERCEPTION project)

- the Symbolic Level (SL) takes the labelled objects as inputs and computes the situations that are likely to be in progress in the environment. The situations are not intended to give a mere representation of the physical features of the objects, but above all, to build a representation of the actions being currently performed by the objects in the environment, so as their temporal linkings.
- the Model of the Environment (ME) records and updates all the relevant elements coming from the Numerical and Symbolic Levels in order to keep a representation of the observed environment. It is based on a dynamic 3D model and preserves the temporal consistency of the moving objects.
- the Resource Management Level (RML) answers the requests of the Symbolic Level for further information, relying on the Model of the Environment and on the Numerical Level. Moreover, RML deals with planning and scheduling of perception actions.

As far as the Symbolic Level is concerned, the models have to be expressive enough to:
• allow numerical data to be translated into symbolic properties;
• take variables into account (the objects to be observed are not known in advance);
• express conditions and constraints on these variables;
• take time into account;
• make predictions about what is likely to be observed;
• send out requests to and get answers from the database (ME) memorizing the observations during the surveillance process;
• represent and deal with symbolic uncertainty.

Moreover, it is important to notice that the applications that are considered do not involve procedures like in [2] or [3] but sequences of actions that are not precisely specified. For instance, somebody wanting to take his car and leave will walk to the car, get into the car, start the car and leave, maybe with several manœuvres; even in a known environment, details on durations, positions, speeds are not available *a priori*.

As far as procedure recognition is concerned, models are defined *a priori*, since, by definition, the real actions have to be carried out in accordance with the procedure (top-down process). On the contrary, models for non-procedural actions have to be defined empirically from the observed reality (bottom-up process).
Consequently, what is proposed in this work is first to design and use basic empirical models representing the expected properties of the observed objects and the expected variations of the properties with time; second, to design a meta-structure on these models so as to deal with the consequence of empirism – or symbolic uncertainty: reality does not match exactly the basic models, basic models may have common features, new models may have to be built.

Sections 2 to 5 deal with the basic models, which are grounded on both a logical language involving 1st order C-cubes and on Petri nets, and with the dynamic scene interpretation reasoning; a complete example is given. Then, the lattice-based meta-structure is explained and exemplified in sections 6 to 8.

2. Knowledge representation for dynamic scene interpretation

Roughly speaking, two main research fields have been dealing with dynamic scene interpretation, Artificial Intelligence on the one hand, and Computer Vision and Signal Processing on the other.

2.1. *Artificial Intelligence approaches*

The Artificial Intelligence field has proposed various frameworks to represent actions, events, plans and intentions for interpretation purposes.
[4] designs a hierarchy of event models, each of them being a template organized around a verb of locomotion, to be matched to scene data. This representation allows deviations from what is expected to be explicitly taken into account.
A formal analysis of actions is given in [5]: an action structure is described as a set of actions that are partially ordered in time, an action being defined by preconditions, postconditions and prevail conditions (holding during action), which is not far from Petri nets.
[6] and [7] perform plan recognition thanks both to a terminological reasoning and a constraint network reasoning based on an action taxonomy and an interval based representation of time.
[8] proposes a model of intentions and beliefs of an agent, a plan being defined as a set of intentions the agent elaborates relatively to pre-existing intentions.
Description logics are proposed in [9] to represent actions for situation recognition; however in [10], it is claimed that terminological systems have to be adapted and extended to meet the needs of image recognition, especially in order to generate and manage hypotheses.

If those approaches propose sound formal frameworks, their main drawback is that they only concentrate on toy problems with restrictive assumptions; for example, actions are supposed to be directly observed, and observations are complete

and certain. No application on real data is shown.

2.2. *Computer Vision and Signal Processing approaches*

In this field, pragmatic approaches have often been proposed for particular problems of recognition in dynamic environments.
In order to detect incidents on an urban round-about from video frames involving known objects, [11] designs three levels of models: kinematical, spatial and relational events corresponding to changes, individual behaviours as results of the propagation of temporal values of the events, and interactive situations as dynamic groupings of objects. The system can detect incidents such as the formation of queues, objects leaving queues, refusal of priority.
Maintaining through perception a coherent interpretation of what is going on is achieved by [2] thanks to situation models, which are sets of event patterns linked with temporal constraints. Recognition is based on a forecast of forthcoming events with their time intervals resulting from a temporal propagation algorithm. The applications are the surveillance of an indoor mobile robot and the monitoring of a complex dynamic system. The approach is quite similar in [3], in which situations models are temporal graphs linking sets of events, each event corresponding to a change in the data. Imperfect matchings between data and models allow incidents to be detected.
Non-temporal and temporal scenarios are also used in [12] for real-time recognition of human activities in a video-surveillance framework.

As a matter of fact, almost all these applications are based on a concept of situation, which is often a temporal network of events, events corresponding to changes that have to be observed in the data coming from the real world. This approach is worth connecting to the notion of spatial network of objects which is used in advanced object recognition systems, such as [13] and [14].

Though the models we propose here are more connected to those kinds of approaches, they intimately mix Artificial Intelligence tools (through 1st order C-Cubes) and discrete Automatic Control tools (through Petri nets), which allows a sound and pragmatic reasoning to be performed for scene interpretation.

3. SL_2 and the SL_2 Interpreted Petri nets for Plan Prototypes

As the Symbolic Level is part of a complete real-word surveillance system, the inputs we consider is what is delivered by the numerical processing of sensor data, that is to say recognized and possibly tracked objects. Therefore the lowest level of knowledge representation for the Symbolic Level is the object level (see also [15] for a different presentation of knowledge representation).

3.1. *SL_2: a symbolic language for the Symbolic Level*

The logical language SL_2 is designed to capture the logical and algebraic conditions that are handled in the symbolic models, in terms of properties and constraints.

SL_2 is a classical 1st-order logic language in which constraints are defined within the same formal frame than the wff. It is based on conjunctions of literals (cubes) associated with constraints, in order to get C-Cubes (Constrained-Cubes). SL_2 is also used as a generic query language to send out requests to and get answers from the Model of the Environment during the surveillance process.

3.1.1. *The terms of SL_2*

$Const$ is the set of *constant* (*ped1*, *V1*), and Var is the set of *variable* symbols $(x, y, t...)$ The variables and constants are used both to denote objects and to represent numerical values. $Fonct$ is the set of *function* symbols which are more specifically devoted to capture numerical operations. Finally the set of all formal terms is the functional closure of the sets of variables and constants by the functions: $Terms = Fonct[Const \cup Var]$.

3.1.2. *The constraints in SL_2*

In this section, simplified formal definitions of constraints [16] are given. Usually, constraints are numerical or boolean, and except for type constraints, they are expressed through binary relations (equations or inequations); consequently, a natural approach is to consider constraints as formal partial orders (fpo).

Let R be the set of all binary relations on a virtual set E. Given two relations r and r', the operators on R are defined as follows: r $\circ$ r' is the composition of r and r'; r^c is the complement relation of r; r $\sqcup$ r' (resp. $\sqcap$) is the join (resp. meet) of r and r'; and r $\sqsubset$ r' is the order relation. The universal bounds are r $\sqcup$ r' = $\top$ (= E $\times$ E), r $\sqcap$ r' = $\bot$ (= $\emptyset$). All the operators are based upon their set equivalent.
As constraints are supposed to be equations or inequations, we define CR, a subset of R, as the set of all the relations that verify the axioms of the partial orders, i.e. reflexivity and transitivity. In order to express strict constraints, the difference ($\neq$) relation (which is not a po) is added to CR (in fact " $\neq$ " = Δ^c where Δ is the diagonal, i.e. the set of all (x, x) couples).
Then the set $Constraints$ of constraints is the functional application of CR on $Terms$:
$Constraints = CR(Terms)$, e.g. $\{v \neq 0, v_0 < v, v < 2a, x + y = z\}$.
Pragmatics:
1- we follow here the classical constraint logic programming approach; in particular it is not necessary to consider logic operators on the constraints (such as: $\{\neg(v \neq 0) \vee ((v_0 < v) \wedge (x + y = z))\}$) as it is always possible to rewrite them, see next note.
2- the semantics of a list of constraints is a conjunction of the elementary constraints belonging to the list. As it is possible to write inequations, the negation operator is intrinsically defined. In the same way, non-linear constraints can be considered.

3.1.3. *Predicates*

Pred is the set of predicate symbols representing logical conditions. An arity function exists from *Pred* onto $\mathbb{N}$ (zero-argument predicates allow zero-order logic properties to be considered): e.g. "armoured" is a zero-arity predicate, whereas "getting-closer-to" has two arguments.

Predicates and constraints make up the standard level of symbolic knowledge processing. The predicates are used to represent the properties of the objects, general properties (permanent knowledge for instance), or conditions linking several objects. Predicates allow the set of atomic formulas, in which the predicates have their correct number of terms, to be built: *armoured*, *getting-closer-to(o2, o1)*, *type(o1, vehicle)*, *speed(o2, x)*, *group(<o2, o1, o3>)*.
NB: at this step, it is formally allowed to consider a negation operator on the atomic formulas ($\neg$*(armoured)*). In fact there is only a very limited part of the data coming from the Numerical Level that leads to define negative information. Furthermore, many supposed negative data are in fact expressed thanks to constraints (*speed(o1,x)* $\wedge(x \neq 0)$). Consequently, negative atomic formulas will not be used widely. However, a special non-standard predicate will be defined in the sequel to express the disappearance of an object.

3.1.4. *C-Cubes*

The last level of SL_2 is the *C-Cubes* (for Constrained-Cubes), noticing that a cube traditionally denotes a conjunction of atomic formulas [17], [18]; as conditions in our symbolic models are combinations of properties and constraints, a C-Cube is a couple (a finite series of atomic formulas, a finite series of constraints), e.g. c $=\{$*(type(o1, vehicle), type(o2, pedestrian), on(o1, parking-lot), speed(o1, x))*, $\{x = 0\}\}$.

3.1.5. *Semantics of C-Cubes*

Let F_n be the File containing the objects delivered by the Numerical Level at time t_n; formulas are supposed to be interpreted at time t_n.

The general semantics of a C-cube is "these formulas are true under these constraints". More precisely, in order to define the semantics of a C-Cube, e.g. $c = (P(x)\ Q(x,y)\ R(x,y,t), \{x = y + z, 0 < y\})$, we have to choose the default quantification of the free variables. This is achieved by applying the existential closure of the C-cubes, i.e. all the variables are supposed to be existentially quantified. The logical constraints are added to the explicit constraints and c becomes: $\exists x, \exists y, \exists z, P(x) \wedge Q(x,y) \wedge R(x,y,t) \wedge \{x = y + z\} \wedge \{0 < y\}$.
Then, the more general rewritten form of c is: $\exists (x, y, z, x', y', x'', t), P(x) \wedge Q(x', y) \wedge R(x'', y', t) \wedge \{x = y + z\} \wedge \{0 < y\} \wedge \{x = x'\} \wedge \{x' = x''\} \wedge \{y = y'\}$.
Consequently, the interpretation of a C-Cube c is made through the file F_n which represents the "universe of the discourse" of the classical model theory of the predicate calculus: $\models c$ *iff*$_{def}\ \overline{\exists}_{F_n} c$.

3.1.6. *The special predicates* Present *and* Missing

It is possible to define a meta-predicate *Present(o)*: *Present(o)* is true *iff*$_{def}$ $(o \in F_n)$. In fact, *Present* may stand as a checking predicate, as it is easy to verify that for any standard predicate $pred(x)$ for which variable x is an argument (e.g. $type(x, T)$), we have: $\forall x \in F_n, pred(x) \rightarrow Present(x)$.

The semantics of the absence of an object is more complex as it is not possible to rely on F_n only. In fact, the concept that has to be captured is the disappearance of an object, which is defined thanks to the predicate *Missing(o)*: *Missing(o)* is true *iff*$_{def}$ $o \in (\bigcup_{i=1}^{n-1} F_i) \perp F_n$.

As a matter of fact, the semantics of the negation is a tricky problem: for instance the predicate $\neg\exists x$ is not universally collectivizing, i.e. it is impossible to determine the set that is supposed to be defined by it outside of a predefined frame. From a strictly logical point of view, $\neg Present$ and $Missing$ are not equivalent ($\models (Missing(o) \rightarrow \neg Present(o))$), but within the frame of the F_n files, their specifications, as they are defined, fit the purposes of the interpretation problem.

3.2. *SL*$_2$ *interpreted Petri nets and plan prototypes*

3.2.1. *Recalls*

A *Petri net* <P, T, F, B> is a bipartite graph with two types of nodes: P={p_1,..., p_i,..., p_m} is a finite set of places; T={t_1,..., t_j,..., t_n} is a finite set of transitions [19], [20]. Arcs are directed and represent the forward incidence function F : P×T → $\mathbb{N}$ and the backward incidence function B : P×T → $\mathbb{N}$ respectively. The matrices associated to F and B are the incidence matrices. An *interpreted Petri net* is such that conditions and events are associated with places and transitions respectively. When the conditions corresponding to some places are satisfied, tokens are assigned to those places and the net is said to be marked.

The choice of Petri nets as a basis for the symbolic models, i.e. plan prototypes, was motivated by the following points:

• they allow sequencing, parallelism and synchronization to be easily taken into account and visualized; therefore, typical or standard behaviours occurring in the observed environment and involving several activities organized in time are naturally represented;

• they are used to monitor, predict (what is going to happen) and review (what happened); prediction is a major element in our case since the activities or behaviours that are likely to be observed next can be forecast, which cuts down the number of hypotheses to be considered by the scene interpretation reasoning;

• more specifically, they have been used in process control and fault diagnosis to represent procedural knowledge [21], system normal and abnormal behaviours and external actions [22], [23], or to implement the diagnosis resaoning itself [24];

• extentions can be considered to cover a wider range of behaviours or to enhance the performance of the scene interpretation reasoning; this point is discussed in

section 7.3.

3.2.2. *SL_2 interpreted Petri nets*

An *SL_2 interpreted Petri net* is an interpreted Petri net in which:
• events associated to transitions are conditions expressed with SL_2;
• actions associated to places are described with SL_2.

3.2.3. *Plan prototypes*

A *plan prototype* is an SL_2 interpreted Petri net in which actions associated to places are *activities*. Activities are described thanks to two sets of SL_2 conditions: a *recognition prototype*, designed for activity recognition; a *carry-out prototype*, designed for activity simulation in time. Both sets *a priori* qualify the objects that are likely to be delivered by the Numerical Level.
Example: A5 *pedestrian-getting-out-of-vehicle*: [recognition prototype *(type(o1, vehicle), type(o2, pedestrian), speed(o2, x), close-to(o2, o1),* $\{x \neq 0\}$)]; [carry-out prototype *(type(o2, pedestrian)* $(pos(o2, t, \vec{X}), pos(o2, t', \vec{X}'), \{\frac{\|\vec{X}' - \vec{X}\|}{t' - t} \leq 6km/h\})$, duration$(d, \{d \leq d_{max}\})$)].

A reachable marking of a plan prototype corresponds to the activities that can hold simultaneously (e.g. A4A6). Therefore a *reachable marking condition* is the conjunction of the SL_2 conditions associated to each place of a given reachable marking. A reachable marking condition must be logically consistent.

An *examination condition* is derived from the SL_2 conditions associated with a set of successive reachable markings. It must be logically consistent.
Examination conditions are characteristic of a given plan prototype and allow that plan prototype to be selected or not during the interpretation reasoning.

An *ending condition* is associated with a sink transition (i.e. a transition with no downstream place) of the SL_2 interpreted Petri net representing a plan prototype (e.g. *missing(o2)*).
An ending condition becoming true allows the interpretation reasoning to consider the activities corresponding to that plan prototype as being terminated.

The different concepts are illustrated on Fig. 2.

4. The Dynamic Scene Interpretation Reasoning

The aim of the reasoning at the Symbolic Level is to output a coherent high level statement of what is going on in the observed environment, from the objects delivered by the Numerical Level. This is achieved through the instantiation of the plan prototypes by the properties of the objects, which in fact is a double problem: the conditions corresponding to activities have to be instantiated, and their temporal consistency has to be checked. A (partially) instantiated plan prototype is called a *P-situation*.
Let $\mathcal{P}$ be the set of plan prototypes.

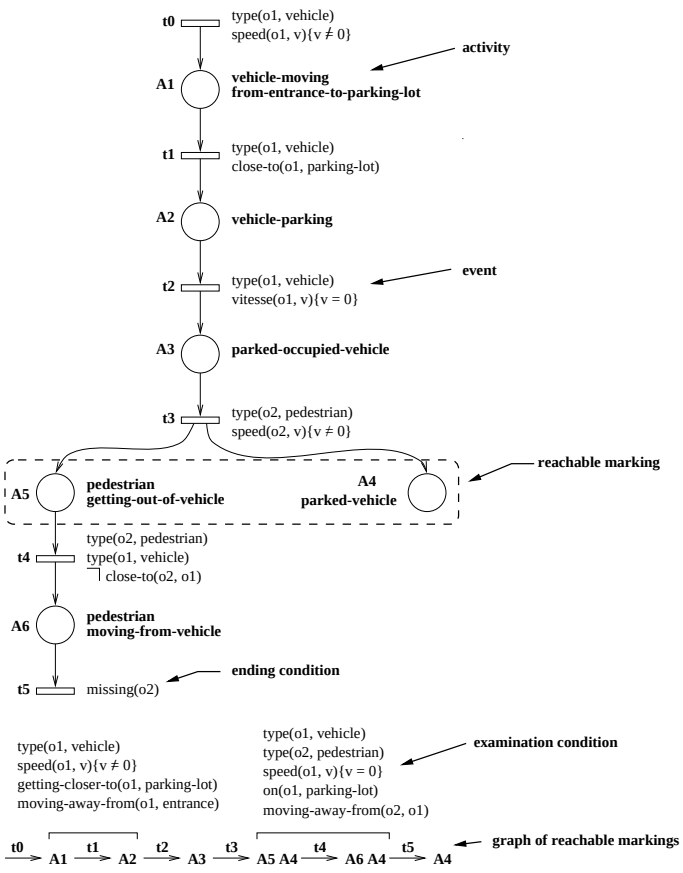

Fig. 2. An SL_2 interpreted Petri net for plan prototype *vehicle-arrival*

4.1. *The algorithm principle*

4.1.1. *Assumptions*

• A0: $\mathcal{P}$ is given *a priori*. It is empirically designed to correspond to what is expected to happen in the observed environment;
• A1: the static and dynamic parts of the scene that are visible for at least one sensor are supposed to be known prior to the interpretation process.
• A2: the objects are not required to be tracked by the Numerical Level (this means that if object *o1* is observed at time t_n and object *o2* at time t_{n+1}, the numerical level may be unable to state that *o2* is the same object as *o1*). In the same way, some objects or properties of objects that are necessary to instantiate some conditions in plan prototypes may not be available. Nevertheless, the Symbolic Level is designed in such a way that a certain robustness of the reasoning is guaranteed, even if

the inputs delivered by the Numerical Level lack completeness (obviously, if object tracks are available, for example as a result of a Kalman filtering, or if all the objects and properties are given, the results of the symbolic processing are enhanced).
• A3: the reasoning is based on the continuity of what happens; therefore a greater importance is given to ongoing P-situations.
• A4: an object is supposed to take part in only one P-situation at a time.

4.1.2. *Matching objects and plan prototypes*

Each time a File F_n containing objects is delivered by the Numerical Level, the Symbolic Level has to answer the question: which conditions in which plan prototype are those objects satisfying?

The link between objects and plan prototypes is created via the marking of the SL_2 interpreted Petri nets: when the properties of an object or a set of objects of File F_n match a reachable marking condition of a given plan prototype P_i of $\mathcal{P}$, an instance of that prototype – a P-situation $(P_{i,m_i,n})$ – is created, with the appropriate marking m_i. This marking evolves as new reachable marking conditions are satisfied.

To answer the question at time t_{n+1}, objects of F_{n+1} are first tried to be explained by the continuation of current P-situations (assumption A3): they are compared to the predictions that are made from the current P-situations i.e., for each P-situation $(P_{i,m_i,n})$, the properties are compared to the conditions associated to the markings $m_i + k$ that can be reached from the current marking m_i:
• if some of the objects match the predictions, the corresponding current P-situations are updated as $(P_{i,m_i+k,n+1})$), i.e. the markings are updated according to the new conditions that are verified; if an ending condition is verified, the corresponding P-situation is said to be terminated.
• if objects do not match the predictions, the corresponding current P-situations are rejected and new plan prototypes are considered, building new P-situations $(P_{j,m_j,n+1})$.
• the final step at time t_{n+1} consists in elaborating *current situations*; a current situation S_{n+1} is a set of coherent P-situations, corresponding to a coherent statement of what is going on, following assumption A4.

Three fundamental processes are therefore involved: new P-situation generation, P-situation continuation checking, current situation building (see Fig. 3).

4.1.3. *New P-situation generation*

New P-situations are generated either at the beginning of the interpretation process, when no plan prototype is instantiated; or when objects do not match the predictions of available P-situations; or when the elapsed time $t_{n+1} \perp t_n$ between Files F_n and F_{n+1} delivered by the Numerical Level is too long in comparison with the scene dynamics.
For each plan prototype P_i of $\mathcal{P}$,

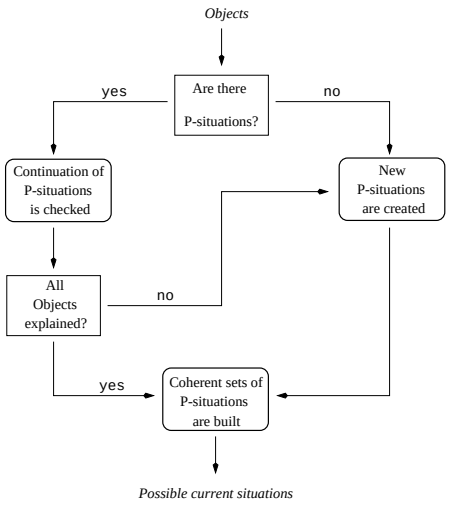

Fig. 3. Algorithm principle

• the objects satisfying the examination conditions are identified; if such objects exist, P_i becomes a candidate to be instantiated as a new P-situation.
• for those objects, the reachable marking conditions they satisfy are identified; if such conditions exist, the plan prototype is instantiated as a P-situation ($P_{i,m_i,n}$) whose marking m_i corresponds to the satisfied conditions.
N.B.: an examination condition or a reachable marking condition generally involves several variables, to be matched with several object identifiers and properties. A File F_n delivered by the Numerical Level at time t_n only contains new information with regard to times $t_i, i < n$. Therefore generally, in order to know if a condition is satisfied, the information of file F_n has to be completed by knowledge that was memorized and updated (thanks to Files F_i or numerical prediction) in the Model of the Environment at time $t_i, i < n$. This knowledge can be reached via the Resource Management Level, through requests and answers expressed in SL_2, e.g.:
if condition to be verified is [*type(o1, vehicle), type(o2, pedestrian), close-to(o2, o1)*], and if *o2* only can be instantiated as ped_2 thanks to file F_n, a request is sent to the Model of the Environment to know if there is a vehicle such that ped_2 is close to that vehicle: [*type(o1, vehicle), close-to(ped_2, o1)*]? The answer may be one or several possible instantiations for *o1*, or "fail".

4.1.4. *P-situation continuation checking*

P-situation continuation is checked when some plan prototypes have already been instantiated as P-situations ($P_{i,m_i,n}$) at time t_n and a new File F_{n+1} is delivered at time t_{n+1} (if elapsed time $t_{n+1} \perp t_n$ is not too long). Functionally, this process includes two phases, prediction generation and object comparison with predictions.

Prediction generation
Given the P-situations ($P_{i,m_i,n}$) available at t_n, the activities that are likely to be observed at t_{n+1} are predicted through both:
• a logical prediction: the list of the markings $m_i + k$ that can be reached from the current marking of each P-situation ($P_{i,m_i,n}$), i.e. the set of the alternate conditions that are likely to be verified at t_{n+1}, is built;
• a temporal prediction: the time that the objects involved in each P-situation ($P_{i,m_i,n}$) at t_n should take to reach the predicted markings $m_i + k$ is estimated; this estimation is mainly based on the carry-out prototypes of the activities.
Comparison of objects and predictions
• semantic comparison: for each P-situation ($P_{i,m_i,n}$), the new objects (delivered at t_{n+1}) that satisfy the conditions corresponding to each predicted marking are searched for; for each of them, the consistency with the objects involved in the marking m_i at t_n is checked (this step is elementary if objects are tracked by the Numerical Level; if they are not, the consistency is checked between the properties of the objects in F_n and the objects in F_{n+1}).
• temporal consistency: the elapsed time $t_{n+1} \perp t_n$ is compared to the temporal prediction. Only the objects and markings corresponding to predicted activities that are consistent with this time consideration are kept.

4.1.5. *Current situations*

Both previous processes result in each object being associated to one or several P-situations with a given marking. The following cases may occur:
• two or more P-situations may in fact correspond to a single one (this is likely to occur if objects are not tracked and only a loose consistency can be obtained between their properties at successive times);
• the same object may appear in several P-situations (this is the case when the object properties match conditions appearing in different plan prototypes).

Given assumptions A3 and A4, coherent sets of P-situations have to be built. These coherent sets, the *current situations*, are the possible interpretations of the dynamic scene, and the outputs of the Symbolic Level.

5. Example

The applied part of PERCEPTION project is devoted to surveillance of a semi-urban environment (a parking-lot). The data collection campaign VIGILE was carried out with a set of heterogeneous sensors (black and white and colour cameras, infrared cameras and a 94 GHz radar) on scenarios that were designed to pose hard problems and to correspond to the operational reality.

The Symbolic Level is implemented as a module of the complete dynamic scene interpretation system. The SL_2 language is implemented in Prolog3®[2] and the interpretation reasoning in C++.

[2] registered trademark of Prologia.

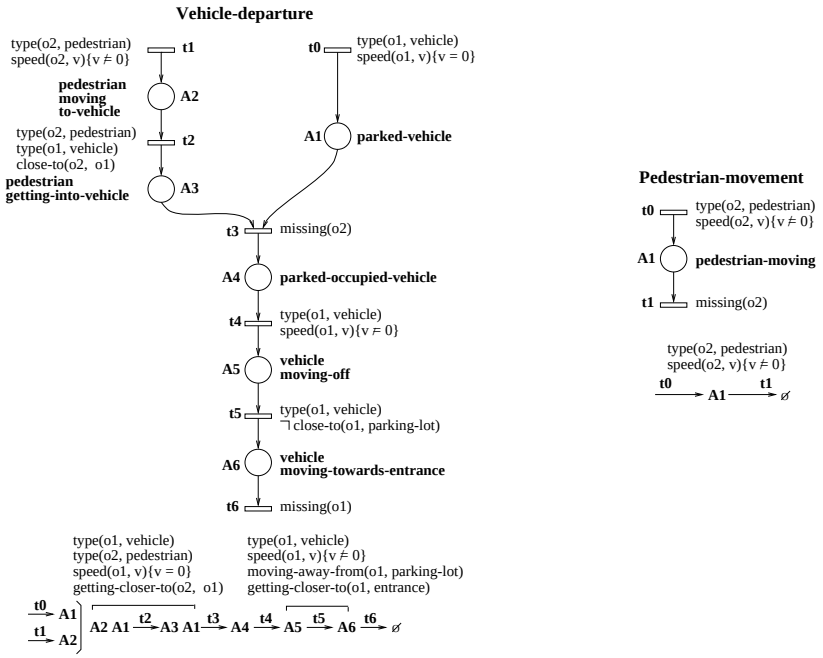

Fig. 4. plan prototype and examination conditions for *vehicle-departure* and *pedestrian-movement*

The example is given on a subset of VIGILE data, coming from a colour wide range camera. For the sake of clarity, let us consider $\mathcal{P}$ as a subset of only three plan prototypes: *vehicle-arrival* (see Fig. 2), *vehicle-departure*, and *pedestrian-movement* (Fig. 4).

The static and dynamic parts of the scene that are visible for the sensor are known prior to the recognition process (assumption A1). Therefore, before F_1 is delivered by the Numerical Level, the Model of the Environment knows the dimensions, positions and speeds (zero-value speeds) of the parked vehicles.

As this example is designed to illustrate the basic matching process of the Symbolic Level, files F_n are supposed to be composed of unambiguous objects, the numerical uncertainties being limited to the scope of assumption A2. The uncertainty issue is tackled in section 6.

• The Numerical Level processes Image-1 and delivers File F_1 at time t_1, with one new object *P1* such that *type(P1, pedestrian), speed(P1, 4km/h)* (a walking pedestrian).

As no current P-situation is available, the Symbolic Level has to select relevant plan prototypes in order to explain the behaviour of this object: therefore the examination conditions of each available plan prototype in $\mathcal{P}$ are tested. As each element of $\mathcal{P}$ involves a moving pedestrian, the set of the potential candidates is $\mathcal{P}$ itself.

Fig. 5. Image-1

Further information is needed in order to output the possible current situations.

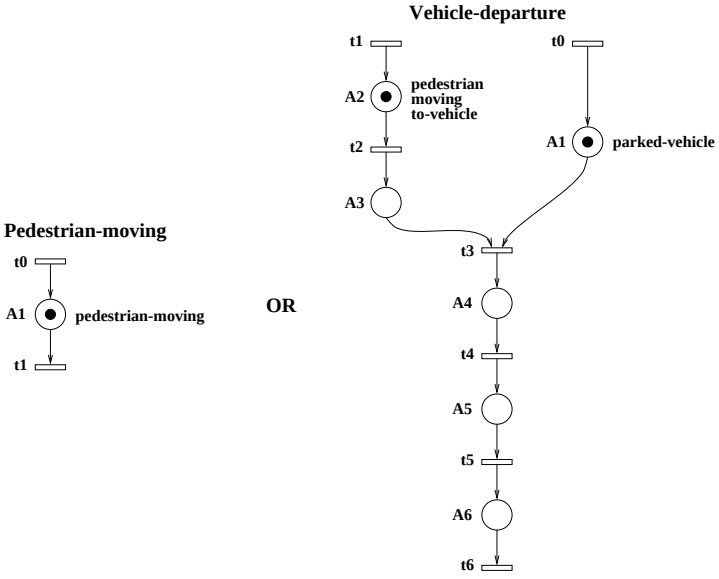

Fig. 6. Current situation S_1

In particular, the examination conditions of plan prototypes *vehicle-arrival* and *vehicle-departure* have to be further checked: is there a parked vehicle such that the pedestrian is moving away from that vehicle? Is there a parked vehicle such that the pedestrian is getting closer to that vehicle? The corresponding requests are sent to the Resource Management Level, which in turn asks the Model of the Environment. The answers are *"fail"* to the first request, and a set of two objects

V1 and *V2* such that *type(V1, vehicle), speed(V1, 0), getting-closer-to(P1, V1)* and *type(V2, vehicle), speed(V2, 0), getting-closer-to(P1, V2)* to the second one.
P-situations can then be built: plan prototype *vehicle-arrival* is rejected, whereas plan prototypes *pedestrian-movement* and *vehicle-departure* are instantiated as P-situations, with the respective markings *** (activity *pedestrian-moving*) and A1A2 (activities *pedestrian-getting-closer-to-vehicle* and *parked-vehicle*) – after checking with the Model of the Environment that the conditions corresponding to activity *pedestrian-close-to-vehicle* are not verified).
The current situation S_1 is shown on Fig.6.

• A new image Image-2 is processed, resulting in File F_2 at time t_2, with $t_2 = t_1 + 4s$: object $P1$ (which has been tracked by the Numerical Level) is now such that *close-to(P1, V1)* (the moving pedestrian is close to one of the vehicles).

Fig. 7. Image-2

The current situation has to be updated: the activity *pedestrian-moving* of *pedestrian-movement* is still verified. But two markings of *vehicle-departure* are now possible: A1A2, corresponding to activities *pedestrian-moving-to-vehicle* and *parked-vehicle*, and A1A3, corresponding to activities *pedestrian-getting-into-vehicle* and *parked-vehicle*. Therefore a P-situation *vehicle-departure* with marking A1A3 is added.
The current situation S_2 is therefore made up of three hypotheses of P-situations:

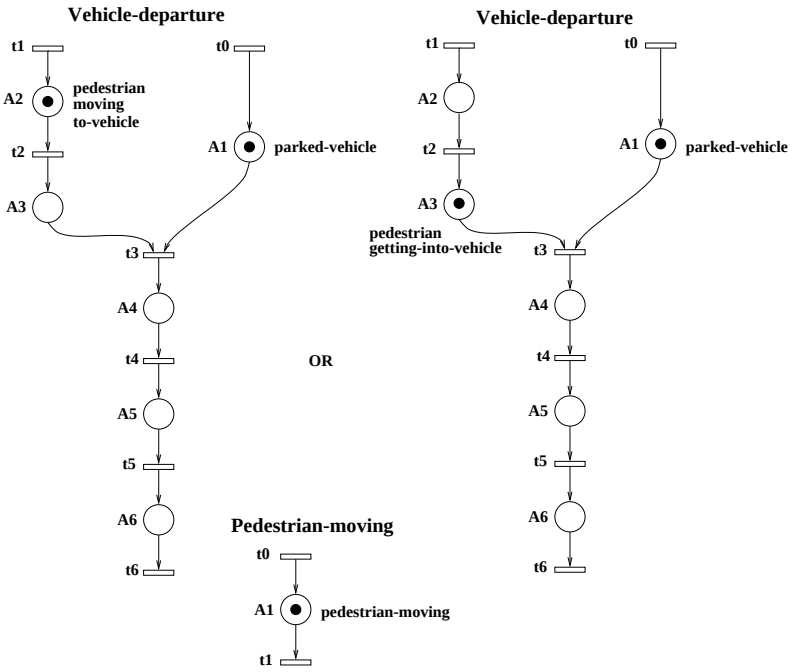

Fig. 8. Current situation S_2

• The next image, Image-3, is processed, resulting in File F_3 at time t_3, with $t_3 = t_2 + 6s$: this File is empty (the pedestrian is not observable).

Fig. 9. Image-3

The Symbolic Level then puts forward five hypotheses (current situation S_3):

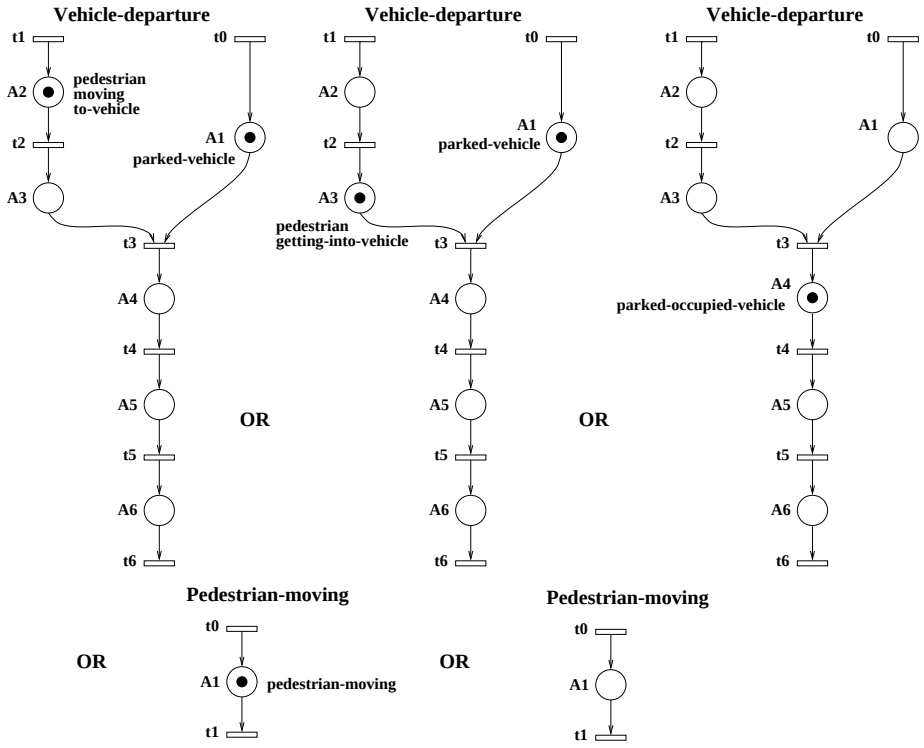

Fig. 10. Current situation S_3

As far as *pedestrian-movement* is concerned, either the activity *moving-pedestrian* is terminated, as the termination condition is verified (and consequently, this P-situation is also terminated), or it is occulted; three markings of *vehicle-departure* are possible: A1A2, with activity *pedestrian-moving-to-vehicle* occulted, A1A3, with activity *pedestrian-getting-into-vehicle* occulted and A4, corresponding to the inobservable activity *parked-occupied-vehicle*.

• The last image is Image-4. File F_4 at time t_4, with $t_4 = t_3 + 14s$ contains object *V1*

(which has been tracked by the Numerical Level), such as now *speed(V1, 10km/h)*, and a new moving object *V2*, such as *type(V2, vehicle), speed(V2, 30km/h)*.
Vehicle *V1* allows the Symbolic Level to go on with P-situation *vehicle-departure*, with activity *vehicle-moving-off* corresponding to marking A5.
As far as *V2* is concerned, a new plan prototype has to be considered: *vehicle-arrival* is a candidate, instantiated as a P-situation with marking A1, corresponding to activity *vehicle-moving-from-entrance-to-parking-lot*.
The final situation S_4 then involves two simultaneous P-situations.

Fig. 11. Image-4

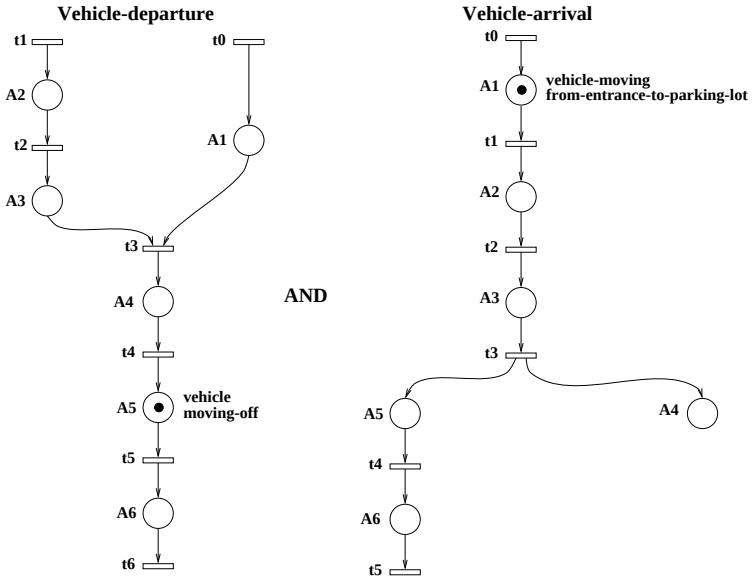

Fig. 12. Current situation S_4

This example shows that, despite basic models are *a priori* designed, non-procedural actions can be recognized, under the strong assumption that objects are perfectly classified and their properties correctly assessed by the Numerical Level. Symbolic uncertainty as a whole is taken into account through alternate P-situation hypotheses.

Nevertheless, the empirical plan prototype grounded theory can be enriched so as to tackle the following problems:
1. objects and their properties are not perfectly recognized,
2. objects and their properties do not match exactly the recognition and carry-out prototypes of the activities,
3. some P-situation hypotheses may be redundant (e.g. in the example, P-situation

pedestrian-movement is "included" in P-situation *vehicle-departure*,
4. most often, what occurs in the environment does not match exactly one of the plan prototypes in $\mathcal{P}$.

Next sections propose a meta-structure for this enrichment.

6. Symbolic uncertainty issues

6.1. *Introduction*

What is searched for is not to propose another interpretation algorithm, but a framework to capture and delimit the symbolic uncertainty and formally characterize the domain of the current situation. Consequently, a formal meta-structure is designed, so as to deal with the four previous points in the following way:

- Points 1 and 2: the uncertainty of the basic information on the one hand (due to the sensors themselves, their positions in the environment and to the numerical processing [25]), and the fact that what occurs in the environment often does not correspond exactly to the recognition and carry-out prototypes of the activities on the other hand, sometimes result in the fact that the properties of some objects cannot be matched to the activities. Nevertheless, a *partial association* could be expected in a lot of cases.
  Example: let us suppose that the recognition prototype of an activity is described with the conjunction of properties $\{a(x) \wedge b(x)\}$. At time t_n, F_n includes an object with properties $a(1)$ and $b(2)$. In a classical data base query, logical expressions $\{a(x) \wedge b(x)\}$ and $\{a(1) \wedge b(2)\}$ are not unifiable: the properties of the object cannot be matched with the activity recognition prototype, even if, obviously, they have things in common. Instead of giving greater importance to one expression or to the other, an intermediate solution can be searched for; for instance, $\{a(x) \wedge b(y)\}$ can at least be expected.

- Points 3 and 4: with an empirically defined set of plan prototypes $\mathcal{P}$, a disjunction of several current situations S_n may be delivered as an output of the Symbolic Level, each of them containing connected P-situations (e.g. they have several activities in common). The idea is to synthesize S_n and give a "parsimonious" result covering all the objects of F_n, as it is proposed in the parsimonious covering theory of [26] for diagnosis. For this purpose, the point is to be able to characterize the result of the "fusion" of several P-situations.

6.2. *A pure symbolic meta-structure*

When numerical data are considered, various tools are available to represent and deal with uncertainty: for example, the notions of mean, variance and standard deviation give a framework within which the uncertain data must be to be considered as non aberrant. The problem is quite different when data are symbolic, in so far as it becomes impossible to state such things as " formula ψ is an approximation of

formula ϕ to the nearest 2%" or to consider "formula $\phi + \delta\phi$"; as a matter of fact, symbolic data are essentially based on discrete frames [27].

Symbolic data may be projected on a numerical space as, in some cases, predefined likelihood or preference measures encode notions such as sensor reliability, information quantity or matching satisfaction. But, as it is a matter of context, no universal method is available [28]. Moreover, these measures automatically define a total order on the pieces of information, which may induce irrelevant relations between elements which were not comparable *a priori*; furthermore, the fusion operators defined from these measures are often purely numerical and produce results in which the symbolic origins and the semantics are lost. For example, let us consider an object O which is recognized by the Numerical Level as a pedestrian, with a confidence level of 0.8. Let us assume that the semantics of that 0.8 is grounded on the fact that a large database of images of pedestrians is used, and that O matches 80% of the database. This 0.8-pedestrian and his properties have then to be matched to activity prototypes, described with symbolic formulas; P-situations, represented by marked interpreted Petri nets, must then be built. The result may be that P-situation1 is recognized with a confidence level of 0.65 and P-situation2 with a confidence level of 0.43, after the application of several numerical aggregation rules. P-situation2 will probably be chosen as the best output. But what is the semantics of those 0.65 and 0.43? What are their justifications? Are there other possible P-situations and what are their characteristics?

Consequently, the approach that is proposed aims at dealing with symbolic uncertainty thanks to purely symbolic tools. It relies on the lattice structure [29], which gives a reasonable improvement of the partially ordered set capabilities as far as non-comparable elements are concerned, while avoiding the strong requirements of totally ordered sets. Moreover, the whole set of the possible solutions can be characterized (and not only the 0.65 or 0.43 ones) and constructive functionalities are intrinsically offered.

The formal meta-structure is now going to be described.

7. Lattices as a basic tool for symbolic uncertainty

7.1. *Recalls*

Given two internal operators $\sqcap$ (infimum) and $\sqcup$ (supremum) on a set E, $(E, \sqcap, \sqcup)$ is a lattice, iff$_{def}$: $\sqcap$ and $\sqcup$ are (L_1) idempotent, (L_2) associative, (L_3) commutative and (L_4) they verify the absorption law $x \sqcap (x \sqcup y) = x$ and $x \sqcup (x \sqcap y) = x$.
Examples: (1) ($\mathbb{N}$, hcf, lcm) is a lattice. (2) If E is a set, the power set of E, $P(E)$, is a lattice with respect to the set union and intersection: $(P(E), \cup, \cap)$. (3) Any totally ordered set is a lattice.
Remark: it is not necessary to have the order relation prior to defining the supremum (the infimum) as the least upper bound (greatest lower bound): L_{1-4} are sufficient to define a lattice properly.

Proposition (consistency): a lattice is an ordered set.
Indeed: the relation $<$ defined on E as: $(x < y) \Leftrightarrow_{def} (x \sqcap y = x)$ is an order relation for which $\sqcap$ and $\sqcup$ respectively represent the greatest lower bound and the least upper bound.

From an informal point of view, the relevance of the lattice approach is the following: let K be the set within which all the symbolic knowledge required to deal with the matching problem is captured. Let us suppose that K is equipped with a lattice structure: when two pieces of information i_1 and i_2 have to be matched, it is possible, even if they are non comparable, to define their infimum $inf(i_1, i_2)$ and supremum $sup(i_1, i_2)$, thus defining a local lattice within which all the possible matching results are included. Hence in the worst case, the matching result is at least as "good" as the infimum and it can be improved so as to reach the supremum or to as be kept at an optimal level. This allows any matching process to be defined as a function M: $M(i_1, i_2) \in [inf(i_1, i_2), sup(i_1, i_2)]$ (*).

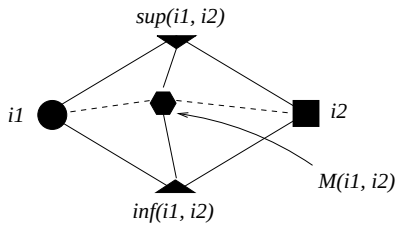

Fig. 13. The local lattice for symbolic matching

This is theoretically justified by the following alternative: either the criteria used to elaborate the matching result are represented within K, hence they verify (*) or else they are not, and then they remain unjustified.

7.2. *Cube lattices*

Let $\mathcal{C}$ be the set of all the logical cubes describing the expected properties (for the sake of clarity, only cubes without constraints will be considered here).

The uncertainty on properties is captured by different means: quantity of information, precision of the terms, logical dependency... For example: {*type(o1,vehicle), speed(o1,x)*} is more informed than {*speed(o1,x)*} for the number of literals is higher; but {*speed(o1,25)*} appears as more informed than {*speed(o1,x)*} for a sake of precision. Unfortunately, the combination of both intuitive criteria is a contradiction: {*type(o1,vehicle), speed(o1,x)*} cannot be consistently compared to {*speed(o1,25)*}.

Sound definitions for the intuitive concepts of union and intersection of two finite information sets are needed, in accordance with the following requirements: the infimum has to capture the common features (while giving more information than the empty set frequently generated by the unification rule); the supremum has to cope with the contradictory criteria: quantity/precision of the information

(while giving a more synthetic result than the set union).

If the definition of the supremum and infimum operators can be supported by the set union and intersection in the propositional calculus frame, first order logic needs more sophisticated tools, especially for unification: we rest not only on the equational theories and the unification theory as defined by [30] or [31], but we also adopt the approach of [32] which allows a lattice on the terms algebra to be defined properly, thanks to the *anti-unification* operator.

Examples: (1) $p(x, g(y, b))$ is the anti-unified literal of $p(a, g(a, b))$ and $p(1, g(b, b))$. (2) The anti-unification of $\{a(x), b(x)\}$ and $\{a(1), b(2)\}$ is $\{a(x), b(y)\}$. In fact, anti-unification allows the infimum to generalize the terms so as to properly enrich the set intersection on the cubes. The anti-unification of two cubes is the union of the anti-unification of all the couples of literals (l_1, l_2) which are built on the same predicate name and such that l_1 belongs to the first one and l_2 to the second one. Conversely, it is essential to reduct the mere set union of cubes in order to define the supremum so as to discard redundancies and to guarantee the verification of L_4. Based upon the same approach than for the anti-unification, the definition of such a reduction relies on the class of the substitutions[3] on terms.

Definition: a cube c is *reducible* if there exists a substitution θ such as $c\theta \subsetneq c$.

Proposition: an irreducible reduction of c always exists and is unique up to variable renaming. It will be noted *reduc(c)*.

Examples: it is clear that $reduc(\{a(x), a(1)\}) = \{a(1)\}$ but $reduc(\{a(1, x), a(y, 2)\}) = \{a(1, x), a(y, 2)\}$; *reduc* looks like the usual factorization but it is more accurate.

If $\mathcal{C}^{\mathbf{r}}$ denotes the subset of all the irreducible cubes of $\mathcal{C}$, it is possible to define constructive operators on $\mathcal{C}^r$:

Definition[4]: let c_1 and c_2 belong to $\mathcal{C}^r$. The supremum and infimum operators $\cup_{\mathbf{c}}$ and $\cap_{\mathbf{c}}$ are defined as:

$\mathbf{c_1} \cup_{\mathbf{c}} \mathbf{c_2} =_{def} reduc(c_1 \cup c_2)$;

$\mathbf{c_1} \cap_{\mathbf{c}} \mathbf{c_2} =_{def} reduc[anti\text{-}unif(c_1, c_2)]$.

Theorem: $(\mathcal{C}^r, \cup_c, \cap_c)$ is a complete lattice.

Proof: as the anti-unification - which was proved to be associative and commutative [32] - is combined with the set operators, L_2 and L_3 are verified for $\mathcal{C}$ itself and consequently for $\mathcal{C}^r$. The proofs of L_1 and L_4 directly come from the unicity of the reduced element (modulo the renaming of variables).

Remark: let $c_1 = \{a(1)\}$ and $c_2 = \{a(2)\}$; if the reduction operator is not applied, $c_1 \cup (c_1 \cap_c c_2) = \{a(x), a(1)\} \neq c_1$ and the absorption law is not satisfied. Such a problem does not occur in $\mathcal{C}^r$ in which $c_1 \cup_c (c_1 \cap_c c_2) = reduc\{a(x), a(1)\} = \{a(1)\} = c_1$.

[3] The classical definition is extended as follows: the substitutions are endomorphisms on terms; given a cube, a substitution is applied to all the terms appearing in its literals.

[4] The algorithms corresponding to these operators are not described in this paper.

Example:

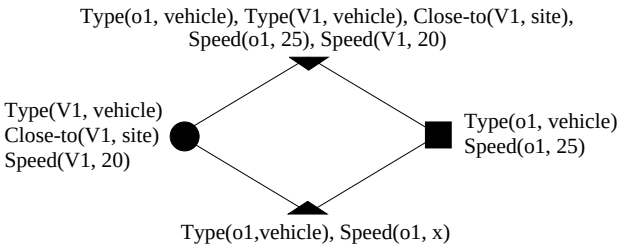

Fig. 14. A cube lattice

Remark: on fig.14, the substitution $\theta = \{o_1 \leftarrow V_1\}$ cannot be applied for the reduction of the supremum because of the apparition of $Speed(V_1, 25)$ which would transgress the inclusion criterion.

Propositions:

• the order relation induced by the lattice structure on $\mathcal{C}^r$ is identical to the set inclusion associated to the instantiation of variables, as follows: $c_1 \leq_c c_2$ iff there exists a substitution θ such that $c_1\theta \subseteq c_2$.

• $(\forall c_1, c_2 \in \mathcal{C}^r)(c_1 \leq_c c_2)$ iff $\models (c_2 \rightarrow c_1)$.

It is worth noticing that the design of the lattice structure on the set of logical cubes does not require the *a priori* definition of a partial order on $\mathcal{C}^r$. Moreover, $\mathcal{C}^r$ is identically ordered by $\leq_c$ and $\leftarrow$; this guarantees the completeness of the approach.

7.3. *Petri net lattices*

As far as P-situations are concerned, would like the supremum of two P-situations to be a more general P-situation combining all the initial information, and the infimum to be a P-situation synthesizing the common elements. Consequently, two different aspects must be considered: the *structure* of the Petri nets – their graphs – on the one hand, and their *interpretations* and their *markings* – the properties – on the other. As the second aspect was dealt with in the previous section, we concentrate on the first one, and define the notions of intersection and union of two Petri nets.

In the literature dedicated to the analysis of the properties of large-scale nets, many approaches based on subnets, sets of common places and transitions, or composition of Petri nets are available [33]. However, these notions are often purely intuitive and hardly defined formally.

Preliminary definitions:

• There exists a *structural automorphism* within a Petri net R iff some rows and columns of the incidence matrices can be permuted without modifying the matrices. This means that some parts of R are structurally equivalent. Consequently we can consider that incidence matrices are defined up to automorphism and that

automorphisms generate internal classes of equivalence within a given Petri net.

• The *upstream and downstream half degrees* of a given node (place or transition) in a Petri net R indicate the number of nodes that are connected to it upstream and downstream respectively.

• A node (place or transition) is *complex* if at least one of its connection half degrees is greater than 1.

Definition: let R_1 and R_2 be two Petri nets. The *intersection* of R_1 and R_2, denoted $R_1 \sqcap R_2$, is the Petri net made up of one or several independent nets resulting from the *compatibility operation* on the set of the maximal nets (in terms of the number of nodes) generated by a one-to-one matching between the nodes of R_1 and the nodes of R_2, up to automorphism, and such as the *preservation of the connection half degrees is maximum.*

The *maximum preservation of the connection half degrees* is a heuristic process allowing the intersection process to focus on complex structural patterns (linear place-transition chains are discarded).

The *compatibility operation* allows the nets of $R_1 \sqcap R_2$ to be independent with regard to their respective images in R_1 and R_2. Isomorphic nets are compatible.

Definition: let R_1 and R_2 be two Petri nets. The *union* of R_1 and R_2, denoted $R_1 \sqcup R_2$, is the Petri net resulting from the connection of R_1 and R_2 on their intersection $R_1 \sqcap R_2$.

Remark: as $R_1 \sqcap R_2$ can appear several times in R_1 and R_2, there may be several possibilities for the union net; as no grounded choice can be made, $R_1 \sqcup R_2$ is taken as the union of those different possibilities.

Example:

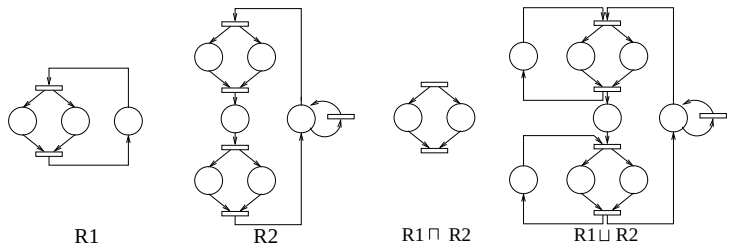

Fig. 15. The Petri net intersection and union

Proposition: let E be the set of Petri nets. (E, $\sqcup$, $\sqcap$) is a lattice [34]. The order relation induced by this structure captures the intuitive notion of the inclusion of two Petri nets.

Example:

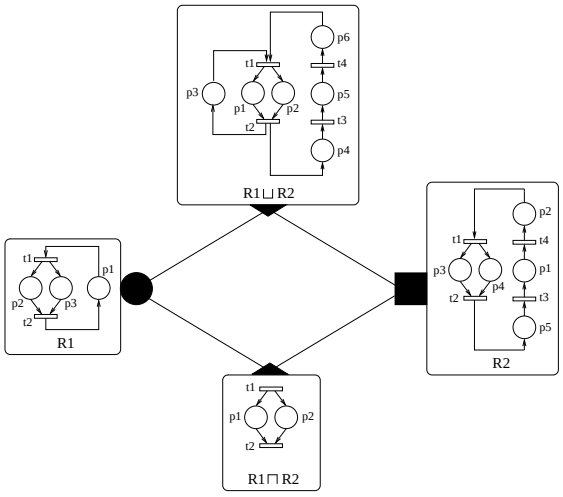

Fig. 16. A four-net lattice

7.4. *Synthesis and example*

The definition of lattice structures on both logical cubes, which represent properties, and Petri nets, which are the basic formalism for plan prototypes, allows the uncertainty issue at the Symbolic Level to be dealt with.

Let F_n be the current File delivered by the Numerical Level at time t_n. The lattice obtained from the cubes representing the properties of the objects on the one hand, and the description of a candidate activity on the other hand, characterizes the *partial matching* between these properties and this description. A particular matching is then *chosen* within the bounds of the lattice, and becomes the activity prototype description in the considered P-situation.

In case the previous step generates several possible P-situations, the lattice calculus on Petri nets allows the different hypotheses to be synthesized within a unique P-situation.

Example: let us consider a scenario involving a moving vehicle, several moving pedestrians and a fire:

Fig. 17. One image from the observed scene

Two plan prototypes have been selected from $\mathcal{P}$:

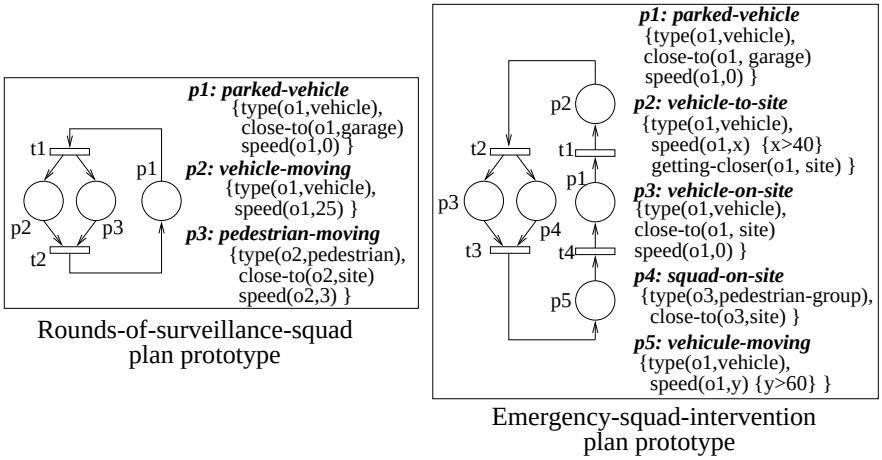

Fig. 18. Plan prototypes for *Rounds of surveillance squad* and *Emergency squad intervention*

At time t_n, the current File F_n is composed of a first object *V1* with properties {*type(V1,vehicle), close-to(V1,site), speed(V1, 20)*} and a second object *GP1* with properties {*type(GP1,pedestrian-group), close-to(GP1, site), speed(GP1, 5)*}. Following section 7.2., F_n partially matches two activities in both plan prototypes; therefore, the current situation S_n is composed of two P-situations P_1 and P_2 (see Fig.19) whose markings correspond to those partial matchings.

Instead of giving the set (P_1, P_2) as a result for S_n – the semantics being it is *either* P_1 *or* P_2, the question is: is it possible to determine a new P-situation synthesizing P_1 and P_2 so as to assess S_n more accurately? Indeed, $\mathcal{P}$ only allows two hypotheses to be set out:

• P_1, rounds of surveillance; indeed the vehicle is moving but there should be only one pedestrian;

• P_2, emergency intervention; indeed there is a group of pedestrians but the vehicle should be parked.

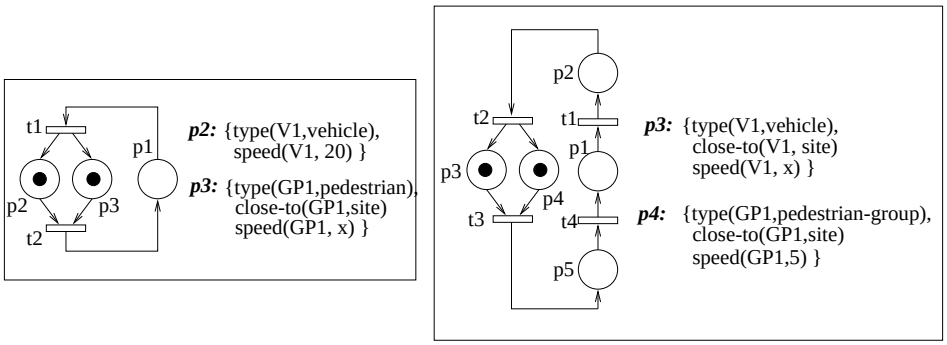

Fig. 19. P-situations P_1 and P_2

The Petri-cube-lattice model allows acceptable solutions to be characterized, and a particular solution can be built as follows: considering the Petri net lattice of Fig.16, where R_1 is the structure of P_1 and R_2 the structure of P_2, $R_1 \sqcup R_2$ can be chosen as the structure for the new P-situation P_*; the marking is chosen within the lattices corresponding to the characterization of the partial matchings of the initial markings of P_1 and P_2 (see Fig.20).

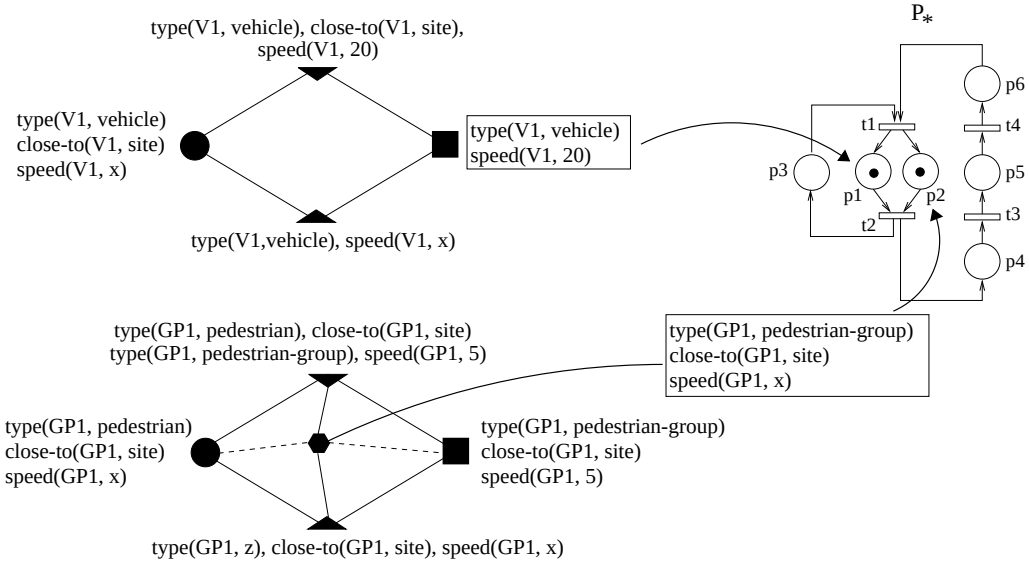

Fig. 20. An example of a synthesized P-situation

It appears that the new P-situation P_* captures a special case of intervention in which, due to the importance of the incident, a group of pedestrians is required while the vehicle is used to evacuate an injured person and pick up additional material. This explains the presence of several pedestrians and the motion of the vehicle.

N.B.: it is essential to notice that the fundamental result of the Petri-cube-lattice model is the *characterization* of the acceptable solutions. Consequently, no algorithm (or fusion operator) to *choose* one particular solution within the set of the acceptable solutions is given here.

8. Conclusion

A formal description of models and of the algorithm for the symbolic interpretation of dynamic scenes has been explained and results delivered by the corresponding implementation on real-world data have been given. Moreover, an algebraic meta-structure has been proposed to cope with symbolic uncertainty with purely symbolic tools: this enables the domain of the acceptable solutions to be characterized, with no *a priori* choice.

Having characterized domains for uncertain properties (expressed as logical cubes) and uncertain plans (described as interpreted Petri nets), it has been shown that "new" results could be built within the domains.

Ongoing research focuses on an extension to the symbolic level of the classical numerical notions of small variation, neighborhood, noise. For instance a pedestrian kneeling down to do up his laces while moving to his vehicle can be considered as a small variation in the P-situation *vehicle-departure*; onlookers watching the squad intervention can be considered as a symbolic noise. The key idea is that those notions are intrinsically included within the lattices that are built for the different pieces of uncertain symbolic knowledge.
Moreover, it allows the information requests that are sent from the Symbolic Level to the Resource Management Level to be more suited to real world uncertainties, in so far as they can be stated in a "smoother" way.

References

[1] Perception. Project int. and final reports. Technical Report 1-2/7995.02-3575.00, Cert, BP 4025, 31055 Toulouse Cedex 04, France, Feb-Oct 1996. In French.

[2] Ch. Dousson, P. Gaborit, and M. Ghallab. Situation recognition: representation and algorithms. In *13th IJCAI*, pages 166–172, Chambéry, France, August 1993.

[3] C. Tessier-Badie, M. Portal, A. Bucharles, Ch Castel, G. Caubet, and Ph. Ferretti. A model-based validation shell for flight data recorders. In *Tooldiag'93*, volume 1, pages 75–80, Toulouse, France, April 1993.

[4] B. Neumann and H.-J. Novak. Event models for recognition and natural language description of events in real-world image sequences. In *IJCAI'83*, pages 724–726, 1983.

[5] E. Sandewall and R. Ronnquist. A representation of action structures. In *AAAI'86*, pages 89–97, 1986.

[6] H. Kautz and J. Allen. Generalized plan recognition. In *AAAI'86*, pages 32–37, 1986.
[7] R. Weida and D. Litman. Terminological reasoning with constraint networks and an application to plan recognition. In *Principles of Knowledge Representation and Reasoning*, Boston, November 1992.
[8] K. Konolige and M. Pollack. A representationalist theory of intention. In *13th IJCAI*, pages 390–395, Chambéry, France, August 1993.
[9] V. Royer. Hierarchical correspondence between physical situations and action models. In *International Workshop on Description Logics*, Roma, Italy, 1995.
[10] B. Neumann and C. Schröder. How useful is formal knowledge representation for image interpretation? In *Workshop on Conceptual Descriptions from Images, ECCV'96*, pages 58–69, Cambridge, UK, April 1996.
[11] J. Thoméré, S. King, S. Motet, and F. Arlabosse. Understanding interactive dynamic situations. In *9th Conference on AI for Applications*, March 1993.
[12] F. Brémond and M. Thonnat. Analysis of human activities described by image sequences. In *10th International FLAIRS Conference*, Florida, May 1997.
[13] F. Sandakly and G. Giraudon. Multispecialist system for 3d scene analysis. In *11th ECAI*, pages 771–775, Amsterdam, 1994.
[14] J. Lemaire. Use of a priori descriptions in a high level language and management of the uncertainty in a scene recognition system (to be published). In *13th International Conference on Pattern Recognition*, Vienna, August 1996.
[15] Ch. Castel, L. Chaudron, and C. Tessier. What is going on? a high level interpretation of sequences of images. In *Workshop on Conceptual Descriptions from Images, ECCV'96*, pages 13–27, Cambridge, UK, April 1996.
[16] J. Jaffar and M.J. Maher. Constraint logic programming: a survey. *The journal of logic programming*, 19,20:503–581, 1994.
[17] A. Tayse, P. Gribomont, G. Louis, and P. Wodon. *Approche logique de l'IA*, volume 1. Bordas, 1988.
[18] L. Chaudron. Lattices for symbolic fusion (in french). In *OSDA'95, International Conference on Ordinal and Symbolic Data Analysis*, pages 135–138, ENST, Paris, 1995.
[19] R. David and H. Alla. *Petri nets and Grafcet*. Prentice Hall, 1991.
[20] T. Murata. Petri nets : Properties, analysis and applications. *IEEE*, 77(4):541–580, 1989.
[21] M. Gallanti, G. Guida, L. Spampinato, and A. Stefanini. Representing procedural knowledge in expert systems: an application to process control. In *IJCAI'85*, pages 345–352, 1985.
[22] J.-F. Dhalluin, R. Gabillard, and M. El Koursi. Application des réseaux de Petri à la commande - contrôle de processus en sécurité. *APII*, 21:531–551, 1987. In French.
[23] Meng Chu Zhou and F. Dicesare. Adaptative design of Petri net controllers for error recovery in automated manufacturing systems. *IEEE SMC*, 19(5), Sept-Oct 1989.
[24] H. Fiorino and C. Tessier. A functional and a behavioural models working together to diagnose failures more accurately. In *AI'94, 14th International Avignon Conference*, pages 301–310, Paris, June 1994.
[25] R.N. Luo and M.G. Kay. Multisensor integration and fusion in intelligent systems. *IEEE SMC*, 19(5):901–931, Sept-Oct 1989.
[26] Y. Peng and J.A. Reggia. *Abductive inference models for diagnosis problem-solving*. Springer-Verlag, 1990.
[27] A. Tarski. The algebra of topology. *Annals of Mathematics*, (45):141–191, 1944.
[28] D. Dubois and H. Prade. Possibility theory and data fusion in poorly informed environments. *IFAC, Control Engineering Practice*, 2(5):811–823, 1994.
[29] G. Birkhoff. *Lattice Theory*. ACM, 1940.

[30] G.P. Huet. Résolution d'équations dans des langages d'ordre 1,2,...,ω. DSc thesis, Univ. Paris VII, 1976.

[31] J.H. Siekmann. Unification theory. *Journal of Symbolic Computation*, 7(3–4), 1989.

[32] J.-L. Lassez, M.J. Maher, and K. Marriott. *Foundations of Deductive Databases and Logic Programming*, chapter Unification revisited. 1987.

[33] Y. Souissi and G. Memmi. Composition of Nets via a Communication Medium. *LNCS 483: Advances in Petri Nets 1990*, pages 457–470, 1990.

[34] C. Cossart, C. Tessier, and L. Chaudron. From Partial Order Operators to Lattice Structures on Generalized Petri Nets. Technical Report 3.760047, Cert, BP 4025, 31055 Toulouse Cedex 04, France, Jan 1997.