Dealing with Temporal Uncertainty and Reactivity in a space mission plan

Thierry Vidal
LGP/ENIT
47, av d'Azereix - BP 1629 - F-65016 Tarbes cedex
thierry@enit.fr

Abstract

In the Deep Space One project, temporal uncertainty in a plan is taken into account through Temporal Constraint Networks in which one further distinguishes between controllable and uncontrollable events. Two properties have been introduced to check the validity of such a plan, namely the Dynamic and the Waypoint Controllabilities. This paper presents a possible way to combine both of them for verifying such a plan before execution, using a game-based strategy based on an automaton formalism, and shows how this tool may model as well more complex reactive behaviours and be used as an execution control tool.

Background and overview

Temporal Constraint Networks (TCN) (Schwalb & Dechter 1997) rely on numerical constraint algebras and are well-suited for modelling a temporal plan (Morris, Muscettola, & Tsamardinos 1998) and check its temporal consistency. In realistic applications the inherent uncertain nature of durations of some tasks must be accounted for, distinguishing between *contingent* constraints (whose effective duration is only observed at execution time) and *controllable* ones (which instanciation is controlled by the agent): consistency must then be redefined in terms of the *Dynamic controllability* (Vidal & Fargier 1999) which encompasses the reactive nature of the solution building process in dynamic domains, checking that a solution can be built in the process of time, each assignment depending only on the previous observations, and still needing to account for all the possible remaining ones. Then *Waypoint controllability* (Morris & Muscettola 1999) means there are some points which can be assigned the same time of occurrence in all solutions, which allows to add wait periods in a plan and hence partition it in more tractable subparts. All that is recalled in next section.

Using a Dynamic controllability checking method through a Timed Game Automaton (third section), the paper shows how in planning this space costly process can be confined in subparts using waypoints, and used as a control execution tool (fourth section). Then some hints are given about using the TGA for modelling more complex reactive behaviours (last section).

Contingent TCN and Controllability

Temporal plans can be represented through Temporal Constraint Networks (Schwalb & Dechter 1997) consisting of time-points (graph nodes) related by constraints (graph edges) that might be mere precedence ($\preceq$) relations, or continuous binary numerical constraints defining the possible durations between two time-points x and y by means of temporal intervals: $l_{xy} \leq (y - x) \leq u_{xy}$ or $c_{xy} = [l_{xy}, u_{xy}]$[1]. A TCN is said to be consistent if one can CHOOSE for each time point a value such that all the constraints are satisfied, the resulting instanciation being a *solution* of the TCN. That can be checked in polynomial time through propagation algorithms.

But one may wish to extend the expressiveness distinguishing between two kinds of time-points: the time of occurrence of an *activated* time-point can be freely assigned by the agent, while *received* time-points are those which effective time of occurrence is out of control and can only be observed. This raises a corresponding distinction between *controllable* and *contingent* constraints (*Clb* and *Ctg* for short): the former can be restricted or instanciated *by the agent* while values for the latter will be provided (within allowed bounds) *by the external world* (see (Vidal & Fargier 1999) for details). For instance, in planning, a task which duration is uncertain and will only be known at execution time will be modelled by a Ctg between the beginning time-point which is an activated one and the ending one which is a received one. The introduction of such uncertainty in the STP results in the following definition of the corresponding *Contingent TCN.*

Definition 1 (CTCN) $\mathcal{N} = (V_b, V_e, R_g, R_c)$ *is a Contingent TCN with*
$V_b = \{b_1, \ldots, b_B\}$: *set of the* B *activated time-points,*
$V_e = \{e_1, \ldots, e_E\}$: *set of the* E *received time-points,*
$R_c = \{c_1, \ldots, c_C\}$: *set of the* C *Clbs,*
$R_g = \{g_1, \ldots, g_G\}$: *set of the* G *Ctgs.*

In the following, a *decision* $\delta(b_i)$ will refer to the effective time of occurrence of an activated time-point b_i, and an *observation* ω_i will refer to the effective duration of the Ctg between x_i and e_i.

[1] This is actually the restricted *STP* (Simple Temporal Problem) where disjunctions of intervals are not permitted.

In a CTCN, the classical consistency property is of no use, since it would mean values for Ctgs can be CHOSEN. The decision variables of our problem are only the activated time-points. And hence a solution should be here, intuitively, an assignment of these activated time-points such that all the Clbs are satisfied, whatever values are taken by the Ctgs. This suggests the definition of some *controllability* property. In (Vidal & Fargier 1999), three different levels of controllability have been exhibited (we will barely focus here on the Dynamic one), completed in (Morris & Muscettola 1999) by the Waypoint Controllability.

Definition 2 (Situations and Schedules) *Given that* $\forall i = 1 \dots G$, $g_i = [l_i, u_i]$,

- $\Omega = [l_1, u_1] \times \dots \times [l_G, u_G]$ *is called the* space of situations, *and*

$\omega = \{\omega_1 \in [l_1, u_1], \dots, \omega_G \in [l_G, u_G]\} \in \Omega$ *is called a* situation *of the CTCN.*

Then, for each time t, *one can define the* current-situation $\omega_{\prec t} \subseteq \omega$ *which is the set of observations prior to* t, *i.e. such that only Ctgs with ending points* $e_i \preceq t$ *are considered.*

- $\delta = \{\delta(b_1), \dots, \delta(b_B)\} \in \Delta$ *is called a* schedule, Δ *being the* space of schedules *(i.e. the cartesian product of interval constraints* $(b_i - b_0)$*)*

Then, for each time t, *one can define the* current-schedule $\delta_{\preceq t} \subseteq \delta$ *which is the sub-schedule assigned so far, s.t.* $\forall b_i \in V_b$ *with* $b_i \preceq t$, $\delta(b_i) \in \delta_{\preceq t}$

In other words, a *situation* is one possible assignment of the whole set of Ctgs, and a *current-situation* with respect to t is a possible set of observations up to time t. And a *schedule* is then one possible sequence of decisions (that might be "controllable" or not), and a *current-schedule* encompasses the notion of reactive chronological building of a solution in plan execution.

Definition 3 (Projection and Mapping)

$\mathcal{N}_\omega$ *is the* projection *of* $\mathcal{N}$ *in the situation* ω, *built by replacing each Ctg* g_i *by the corresponding value* $\{\omega_i\} \in \omega$. *In (Vidal & Fargier 1999) a projection is proved to be a simple TCN corresponding to a STP.*

μ *is a mapping from* Ω *to* Δ *such that* $\mu(\omega) = \delta$ *is a schedule applied in situation* ω.

Intuitively, a CTCN will be "controllable" if and only if there exists a mapping μ such that every schedule $\mu(\omega)$ is a solution of the projection $\mathcal{N}_\omega$. In fact this is only the "weakest" view of the problem (called Weak controllability in (Vidal & Fargier 1999)), that assumes that one knows the complete situation before choosing one schedule that will fit. But when a plan is executed, decisions are taken in a chronological way, and for each atomic decision the agent knows the past observations, but the observations to come are still unknown. The Dynamic controllability property defined below takes that into account.

The Waypoint controllability has been further introduced in (Morris & Muscettola 1999), stating that there are some points for which all the schedules share the same time of occurrence, whatever the situation is[2]. Those waypoints serve as "meeting" time-points in a plan, when the agent waits for all the components of a subpart of the plan to be over before starting the next stage. Waypoints can be created during the planning process through the addition of "wait periods" (Morris & Muscettola 1999).

Definition 4 (Dynamic/Waypoint controllability)

- $\mathcal{N}$ *is* Dynamically controllable *iff*
  (1) $\exists \mu : \Omega \longrightarrow \Delta$ *s.t.* $\mu(\omega) = \delta$ *is a solution of* $\mathcal{N}_\omega$
  (2) $\forall (\omega, \omega') \in \Omega^2$, *with* $\delta = \mu(\omega)$ *and* $\delta' = \mu(\omega')$, $\forall t$, *if* $\omega_{\prec t} = \omega'_{\prec t}$ *then* $\delta_{\prec t} = \delta'_{\prec t}$
- $\mathcal{N}$ *is* Waypoint controllable *with respect to* $W \subset V_b$ *iff*
  (1) $\exists \mu : \Omega \longrightarrow \Delta$ *s.t.* $\mu(\omega) = \delta$ *is a solution of* $\mathcal{N}_\omega$
  (2) $\forall (\omega, \omega') \in \Omega^2$, *with* $\delta = \mu(\omega)$ *and* $\delta' = \mu(\omega')$, $\forall x \in W$, $\delta(x) = \delta'(x)$

Dynamic and Waypoint controllability are proven to be exponential in the number of time-points. But the complexity of Waypoint controllability is actually exponential in the maximum number of time-points between two waypoints (Morris & Muscettola 1999), which means the more waypoints one considers in the CTCN, the less complex is Waypoint controllability.

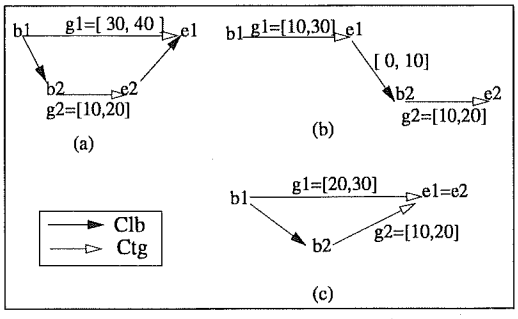

Figure 1: Illustrating controllability

Figure 1 exhibits a threefold example. In the first case (a), one task is constrained to occur *during* the other[3]. For instance that might be a data sending task to an orbiter for a planetary robot that needs to be achieved during a visbility orbiter temporal window, both durations being uncertain. The second example (b) shows two successive contingent tasks, with a maximum (controllable) delay of 10 time units between them. For instance a planetary rover might have to send data within 10 seconds after having correctly directed its antenna towards the orbiter. The third example (c) shows two tasks, one activated after the other, that should finish exactly at the same time, though they

[2] We have chosen to restrict in some sense the original Waypoint controllability definition that allowed the set W to contain received events, since in general those would not satisfy the Waypoint controllability property.

[3] Unlabelled arrows stand for simple precedence, that should simply be $[0, +\infty[$ intervals in the TCN framework.

both have uncertain durations. Obviously this last example cannot be accepted as a valid plan, since no execution can guarantee that the constraint will be met. Interestingly enough, the three examples are all controllable in the weak sense (i.e. condition 1 holds), but only examples (a) and (b) meet condition 2 of Dynamic controllability, which fits what one should expect in realistic planning. Moreover, in example (a), one should get that b_2 must be activated at most 10 time units after b_1 to ensure the satisfaction of the relation whatever values are taken by ω. Last, about Waypoint controllability, in example (b), b_2 shouldn't be set as a waypoint since the time at which one can release the second task depends upon the effective duration of the first one, and hence cannot be set in advance. In (Morris & Muscettola 1999), Waypoint controllability is proven to be equivalent to Dynamic controllability under some restrictive conditions (see page 5), that unfortunately do not hold in case (c) ($W=\{b_1\}$ satisfies Waypoint controllability: actually it lets b_2 depend on the outcome of ω_1 and ω_2), hence Dynamic controllability still has to be checked.

Using Timed Game Automata

The method we present here relies on the *timed automata* model used for describing the dynamical behaviour of a system (Alur & Dill 1994). It consists in equiping a finite-state automaton with time, allowing to consider cases in which the system can remain in a state during a time T before making the next transition. This is made possible by augmenting states and transitions with "continuous variables called *clocks* which grow uniformly when the automaton is in some state. The clocks interact with the transitions by participating in pre-conditions (*guards*) for certain transitions and they are possibly reset when some transitions are taken." Guards may be converted in *staying conditions* (Asarin, Maler, & Pnueli 1995) in states.

Such tools are well-suited (Asarin, Maler, & Pnueli 1995) to *real-time games* for controlling reactive systems, where transitions are divided in two groups (such as constraints in CTCN) depending on which of the two players control it: the *controller* or the *environment* ("Nature") and some states are designated as *winning* for the controller. This extension of the classical discrete game approach, with has the following features: (1) there are no "turns" and the adversary need not wait for the player's next move, and (2) each player not only choose between alternative transitions, but also between *waiting* or not before taking it". Finding a *trajectory* (i.e. a path in the automaton) reaching a winning state defines a so-called *safety game* policy. We propose hereafter our own definition of a Timed Game Automaton that follows those lines and will perfectly fit our purpose.

Definition 5 (Timed Game Automaton)
$\mathcal{A} = (\mathcal{Q}, \Sigma, \Gamma, \mathrm{S}, \mathrm{T})$ *is a timed game automaton (TGA) where*

- $\mathcal{Q}$ *is the discrete and finite set of states* q_i*, with three special cases:*
  - q_0 *is the initial state,*
  - $q_o k$ *is the unique* winning *state,*
  - $q_\dagger$ *is the unique* losing *state;*
- $\Sigma = \Sigma_b \times \Sigma_e$ *is the input alphabet such that any label in* Σ_b *is of the form* b_i *and any label in* Σ_e *is of the form* e_i*;*
- $\Gamma = \Gamma_{sw} \cup \Gamma_{et}$ *is the discrete and finite set of clocks and may be of any cardinality (i.e. one can define as many clocks as one needs), where* Γ_{sw} *is the set of* clocks*, on which are defined two sets of conditions and actions:*
  - $\mathtt{Re} = \{(sw_i \leftarrow 0)\ s.t.\ sw_i \in \Gamma_{sw}\}$ *is the finite set of all possible clock reset functions,*
  - $\mathtt{Gu} = \{(l_i \leq sw_i \leq u_i)\ s.t.\ sw_i \in \Gamma, (l_i, u_i) \in \mathbb{Z}^2\}$ *is the infinite set of all possible clock conditions that will be used as guards or staying conditions.*
- $\mathrm{S} : \mathcal{Q} \rightarrow \mathtt{Gu}$ *assigns staying conditions to states;*
- $\mathrm{T} = \mathrm{T}_b \cup \mathrm{T}_e \subseteq \mathcal{Q}^2 \times \Sigma \times \mathtt{Gu} \times \mathtt{Re}$ *is the set of transitions of the form*

$\tau = < q, q', \sigma, g, r >$ *with a distinction between*
- $\tau \in \mathrm{T}_b$ *is an* activated *transition* iff $\sigma \in \Sigma_b$*,*
- $\tau \in \mathrm{T}_e$ *is a* received *transition* iff $\sigma \in \Sigma_e$*.*

For an activated transition, the controller decides the time of activation by "striking" the clock at any time within the two bounds of the guard, while a received transition will be taken at some unpredictable time within the bounds.

(Vidal 2000) provides a translation algorithm from a CTCN to a corresponding TGA: an event in $\mathcal{N}$ will appear as a translation labelled with this event in $\mathcal{A}$. Similarly, temporal intervals in $\mathcal{N}$ will appear as guards in $\mathcal{A}$. The example of Figure 1(a) gives the corresponding TGA in Figure 2, where on each transition one can view the guard condition and the label above and the clock reset below. Two clocks g_1 and g_2 are used by analogy with the corresponding contingent durations. One can notice that the system is not a priori prevented to receive e_1 before e_2, which would violate a constraint, hence receiving e_1 before e_2 appears as a transition to the losing state in the automaton.

Then (Vidal 2000) provides a *synthesis* algorithm that checks Dynamic controllability, using *controllable predecessors* operator that recomputes from a state q the previous ones, revising the staying conditions so as ensure the losing state cannot be reached any longer. The operator is recursively applied from the initial set of winning states until it reaches a fixed point. If q_0 is in the final set, then the controller can always win the game. This is illustrated in Figure 2: in q_2, $g_1 \leq 30$ is added (and hence the transition to $q_\dagger$ can be removed) and propagated backward to q_1: considering that one may stay up to 20 time units in q_2 because of the uncontrolled clock g_2, then g_1 shouldn't get to more than $30-20=10$ time units in q_1. This condition corresponds to a restriction of the Clb (b_2-b_1) in the original CTCN.

Modifying the example by replacing the second Ctg by $g_2 = [25, 35]$ would as one should expect it lead the

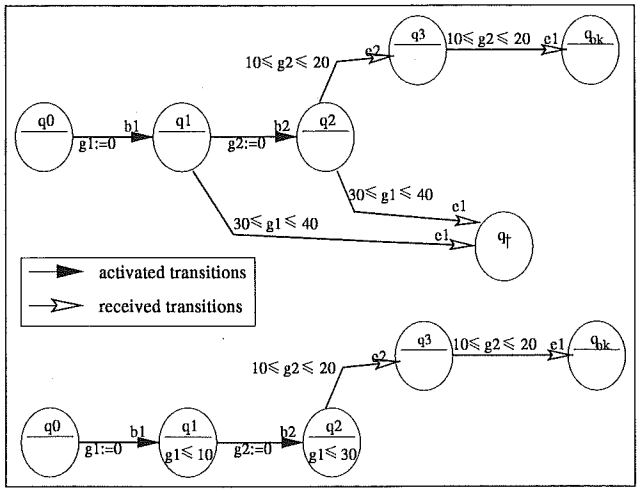

Figure 2: TGA interpretation of a CTCN

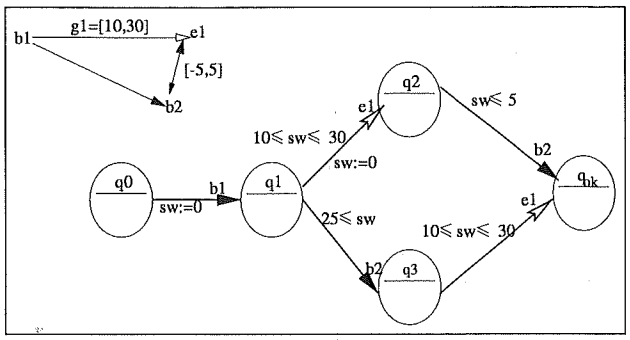

Figure 3: An example of reactive behaviour

synthesis algorithm to fail, since adding the staying condition $g_2 \leq 30$ in q_2 would not propagate back to q_1 considering that one may stay in q_2 up to 35 time units. Hence one would get as a result that the CTCN is not Dynamically controllable.

On the application of TGA in planning

Relevance and conditional nature of the approach

Expressing and reasoning upon temporal uncertainty is needed in most of planning and scheduling applications such as the NASA project Deep Space One (Morris, Muscettola, & Tsamardinos 1998; Morris & Muscettola 1999), where having to deal with contingent durations means basing activation decisions on previous observations, as in Figure 1(b).

A more interesting example is the following. Let us suppose the on-board computation system predicted that a meteorit (that has been detected by a radar) will pass at a minimal distance from the spatial vehicle between 10 and 30 seconds from now. "Now" is the time-point b_1, and the radar will issue event e_1 when the meteorit is at minimal distance (i.e. when the distance starts increasing again). The automatic planning system is given the goal to take a picture of the meteorit close enough to this ideal position, i.e. within 5 seconds before or after e_1. This decision is modelled by time-point b_2. Besides one may easily suppose that the picture should be taken the sooner the better, after what the spatial vehicle is planned to change direction to escape the area in which other meteorits are expected ... Figure 3 shows the resulting synthesized TGA: one shouldn't decide to activate b_2 less than 25 seconds after b_1, but if e_1 is received before, then just activate b_2 within 5 seconds. This is Dynamically controllable, but here one gets two completely different schedules depending on ω_1 being lower or greater than 25, which corresponds to some kind of conditional planning. Moreover, the TGA models a reactive opportunistic behaviour, since the activation of b_2, initially scheduled at 25, might be opportunistically made before on early reception of e_1.

Hence the TGA implicitely models reactive behaviours and conditional plans, since two transitions from the same state correspond to a branching (OR node), while in a CTCN time-points are AND nodes. Therefore, the CTCN formalism is a very powerfull tool for describing the *specifications* of a dynamic system (as e.g. a contingent plan), through the constraints it has to meet, expressing rich temporal information in a compact way. Whereas the TGA is a *simulation* model that captures all the possible *execution scenarii* of the plan: it has the advantage of providing efficient and robust techniques to check its "safety", but runs the risk of combinatorial explosion in the number of states.

The TGA as a plan execution control tool

Another strength of the TGA is that it can be directly used as an execution supervisor tool, since the synthesis not only checks Dynamic controllability but also gives schedules of starting times for the planned tasks. Moreover, in constraint networks only a deterministic sequence of decisions is issued, whereas with a TGA one can get a reactive execution supervisor, with disjunctive possible trajectories, which makes it well adapted to stochastic environments.

Complexity and practical efficiency issues

As far as complexity is concerned, the algorithms for building and synthesizing the automaton are respectively linear and in logarithmic time in the number of states, which is in the worst case p^B, where B is the number of activated time-points and p the *degree of parallelism* (i.e. the maximum number of time-points possibly occurring at the same time) (Vidal 2000). Hence the complexity of the approach lies in the possibly exponential number of states developped, which depends upon the network feature p.

This potential combinatorial explosion can be restricted when the plan is mostly a sequence and there are not so many events in parallel, which is often

the case in space mission planning. Moreover, one could use *dispatchable* networks (Morris, Muscettola, & Tsamardinos 1998), that are TCNs in which redundant constraints are removed and only minimal paths are exhibited, so as to optimize propagation during execution. This could restrict as well the number of states produced in the TGA. Besides, automata-based techniques might be improved to reduce the number of states produced, considering that two subsequences containing the same set of events, though in distinct orders, might converge on the same state, or using more complex abstraction views (see (Asarin, Maler, & Pnueli 1995)).

One may also accept an incomplete checking algorithm in the long term (based on incomplete propagations in the CTCN), using the TGA only in the short term, as far as execution runs: the algorithm anticipates the possible losing state deadends and can activate any necessary recovery action in advance.

But the most promising idea seems to be the one that follows ...

An extended framework using TGAs and waypoints

It is argued in (Morris & Muscettola 1999) that choosing cleverly the set of waypoints through addition of "wait" periods in the plan might lead to Dynamic controllability being equivalent to Waypoint controllability. Anyway, trying to design a plan in this way might lead to a high number of waypoints lowering the plan optimality: an opportunistic scenario like the one in Figure 3 would not be possible. In other words, one may compel the executed plan to "play for time" at some points when it is not really needed.

But waypoints might be used merely to restrict Dynamic controllability checking in all subparts of the networks between any pair of waypoints: we need to prove the following property, where $\mathcal{N}_{(}x, x')$ stands for the restriction of $\mathcal{N}$ to the time-points temporally constrained to be after x and before x'.

Property 1 (Partitioned controllability) *$\mathcal{N}$ is Dynamically controllable iff*

(1) $\mathcal{N}$ is Waypoint controllable wrt $W \subset V_b$, and

(2) $\forall (x, x') \in W^2 s.t.\ x \preceq x'$ and $\nexists x''/x \preceq x'' \preceq x'$, $\mathcal{N}_{(}x, x')$ is Dynamically controllable

Sketch of proof. The proof is rather straightforward. Dynamic controllability means that a current-schedule will only depend on the corresponding current-situation. For any time t between two waypoints x and x', $\delta(x)$ is set in all schedules, which means it does not even depend on $\omega_{\preceq \delta(x)}$, and any forthcoming decision will not depend on it either. Consequently, $\delta_{\preceq t}$ does only depend on the part of the situation between $\delta(x)$ and t, which is equivalent to the formulation above. $\diamond$

Then, a possible global algorithmic framework to check Dynamic controllability could be to (1) use some heuristic (to be defined at the planning engine level) to decide to introduce wait periods or not while planning, and (2) check Dynamic controllability by building a TGA between any pair of successive waypoints[4]. The idea is to introduce "not so many" waypoints in the plan, so as to still meet in one hand high optimality requirements for the plan, while on the other hand drastically reducing time and space complexity of the TGA approach by only synthesizing automata corresponding to subparts of the whole plan. This decomposition technique hence offers a nice trade-off between expressiveness, optimality and efficiency.

The full expressiveness of TGAs in planning

The expressive power of TGAs is obviously larger than what is used here, and might be of interest for other planning problems.

Generalized conditional planning

A first obvious remark is about the relation between the TGA model and conditional or reactive planning: if the TGA allows to represent the inherent conditional nature of a CTCN, then why not using it for different kinds of branching in planning ? For instance, consider information gathering problems, in which a perception action is included in a plan, and the next steps of the plan depend on the outcome of this action. Or more generally speaking, all cases in which non-determism of actions has to be dealt with. If one wishes to represent distinct evolutions of a plan, then this corresponds to some disjunctions (OR nodes) that are naturally represented in a TGA and may be smoothly merged with temporal contingency branching. Hence TGA might be used in such cases as well. The only difference is that this kind of non-determinism cannot be represented in a compact way in a CTCN, and one can hardly avoid the use of OR nodes in addition to partial order (i.e. AND nodes) in a temporal constraint-based planning graph.

Preprocessed plans and reactive planning

Sometimes a unique plan with branching nodes is not enough to solve a problem. One may need to use a library of subplans to run in reaction to typical events. For instance a planning system may produce a nominal plan together with a number of alternative "recovery" sequences to be runned in replacement when a modeled disturbance occurs, as in (Washington, Golden, & Bresina 1999). Instead of having those subsequences connected to a node of the nominal plan, they may be stored in a library and connected to a type of received and unpredictable event, which defines a more general kind of uncertainty than the one addressed in this paper (not only the time of occurrence of the event is unknown, but the occurrence of the event itself). Receiving this event will force the execution supervisor to

[4]Wait periods will by construction introduce waypoints such that the network is necessarily Waypoint controllable, which hence does not need to be checked.

temporally abandon the current plan to run the corresponding recovery sequence.

Then a more general reactive framework needs to be designed, as in (Vidal & Coradeschi 1999): one can define temporal *chronicles* corresponding to each "abnormal" scenario, with the possibility of having several "faulty" events in elaborated and rather complete scenarios. A purely reactive TGA framework can be designed, where on-line automaton building is processed, in reaction to received events: the system dynamically matches the received event with the chronicles that contain it, and synthesizes the possible next steps in those chronicles in one unique incremental automaton.

By mixing the general off-line planning framework presented in this paper with this purely reactive behaviour, one get a real-time planning system that might be very robust in stochastic environments.

Synchronization constraints

Last, sometimes so-called synchronization constraints that are outside the scope of classical temporal algebras may be useful. Three of them have been defined in (Fargier *et al.* 1998):

Parmin Two "tasks" A and B are related by a parmin if both starts at the same time and the first that is finished terminates the other as well (e.g. task of recording a given sequence of signals from a meteorit and a task waiting for the radar to notify that the meteorit has got out of sight);

Parmaster This is the same as parmin, except that only the first "task" in the relation forces the second one to finish at the same time (e.g. the previous example with the length of the recording sequence not predetermined);

Parmax Two "tasks" A and B are related by a parmax if both starts at the same time and the first that is finished has to wait for the second one to finish as well before next steps in the plan are processed.

The two first ones are interruption-like behaviours, while the third one is an "appointment" constraint. The CTCN model cannot model these constraints: a parmin for instance implies a ternary constraint (Fargier *et al.* 1998) between (1) the expected end of the first task, (2) the expected end of the second task, and (3) the "effective end" of both, that will actually be one of the two previous ones (we recall that in CTCNs only binary constraints are allowed). But interestingly enough, those behaviours are implicitely conditional behaviours, and are very easy to model through a TGA.

Conclusion

This paper has brought to light the advantage of using Timed Game Automata for checking dynamic temporal properties of a plan in the presence of temporal uncertainties. Discussing efficiency and usefullness in practice, it suggests the addition of heuristically and sparingly selected wait periods in the plan to partition it so as to be able to check the Dynamic controllability property locally. The applicability of Timed Game Automata to more general conditional and reactive planning features may as well allow to address more realistic real-time planning problems in stochastic environments.

Acknowledgement

The author is grateful to Paul Morris (NASA Ames Research) for fruitful discussions and his suggestion of the modified Dynamic controllability definition.

References

Alur, R., and Dill, D. 1994. A theory of timed automata. *Theoretical Computer Science* 126:183–235.

Asarin, E.; Maler, O.; and Pnueli, A. 1995. Symbolic controller synthesis for discrete and timed systems. In Antsaklis, P.; Kohn, W.; Nerode, A.; and Sastry, S., eds., *Hybrid Systems II, LNCS 999*. Springer Verlag.

Fargier, H.; Jourdan, M.; Layaïda, N.; and Vidal, T. 1998. Using temporal constraint networks to manage temporal scenario of multimedia documents. In *ECAI-98 workshop on Spatio-Temporal Reasoning.*

Morris, P., and Muscettola, N. 1999. Managing temporal uncertainty through waypoint controllability. In Dean, T., ed., *Proceedings of the 16th International Joint Conference on A.I. (IJCAI-99)*, 1253–1258. Stockholm (Sweden): Morgan Kaufmann.

Morris, P.; Muscettola, N.; and Tsamardinos, I. 1998. Reformulating temporal plans for efficient execution. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR-98).*

Schwalb, E., and Dechter, R. 1997. Processing disjunctions in temporal constraint networks. *Artificial Intelligence* 93:29–61.

Vidal, T., and Coradeschi, S. 1999. Highly reactive decison making: a game with time. In Dean, T., ed., *Proceedings of the 16th International Joint Conference on A.I. (IJCAI-99)*, 1002–1007. Stockholm (Sweden): Morgan Kaufmann, San Francisco, CA.

Vidal, T., and Fargier, H. 1999. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental & Theoretical Artificial Intelligence* 11:23–45.

Vidal, T. 2000. Controllability characterization and checking in contingent temporal constraint networks. In *Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR2000).* Breckenridge (Co, USA): Morgan Kaufmann, San Francisco, CA.

Washington, R.; Golden, K.; and Bresina, J. 1999. Plan execution, monitoring, and adaptation for planetary rovers. In *IJCAI-99 workshop on Scheduling and Planning meet Real-Time Monitoring in a Dynamic and Uncertain World*, 9–15.