Multi Agent Cooperative Robotics Laboratory

Kansas State University
234 Nichols Hall
Manhattan, KS 66506-2302

Phone: (785) 532-6350
Fax: (785) 532-7353
http://macr.cis.ksu.edu/

Technical Report

Analyzing GMoDS Goal Models using Petri Net Semantics

By

Scott A. DeLoach

MACR-TR-2010-03

May 25, 2010

Table of Contents

Analyzing GMoDS Goal Models using Petri Net Semantics

Scott A. DeLoach

Department of Computing & Information Sciences
Kansas State University
Manhattan, Kansas, USA
sdeloach@k-state.edu

Abstract. This report presents a formalized of the semantics for the Goal Model for Dynamic Systems in terms of Colored Petri Nets.

1 Introduction

As the multiagent system approach has evolved, several key aspects are being viewed as useful elements of future complex and adaptive systems, regardless of whether or not they use the term agent or multiagent systems. One of these keys is the goal-driven approach used in intelligent agent systems. Not only do goals allow agents to act autonomously, they also provide a mechanism for describing *what* a system should do without tying it to *how* it should be accomplished.

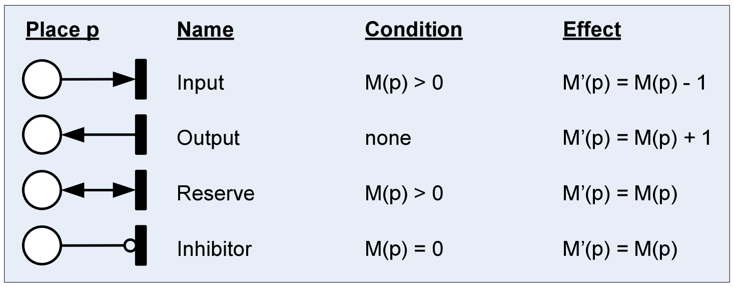
Unfortunately, we were unable to find a general framework for developing complex systems that used goals to capture requirements while using the same set of goals to drive the system at runtime. Not only would such a framework allow systems to adapt to the dynamics of the problem being solved and the system's environment, it would also provide a consistent view of the requirements from analysis, through design to implementation. Consequently, we have spent significant time at the Multiagent and Cooperative Robotics Laboratory developing a goal-modeling framework called the Goal Model for Dynamic Systems (GMoDS). GMoDS provides a set of models for capturing system level goals and for using those goals during design and at runtime to allow the system to adapt to dynamic problems and environments. In a GMoDS, goals are organized in a goal tree such that each goal is decomposed into a set of subgoals using an OR-decomposition or an AND-decomposition. GMoDS models include the notion of goal *precedence* and goal *triggering* [1]. A *precedes* relation determines which goals must be achieved while a *trigger* relation signifies that a new goal may be instantiated when a specific event occurs during the pursuit of the goal. More detail about GMoDS can be found in [1].

The benefits of having a single goal model used for both requirements specification and at runtime include the ability to perform analysis of runtime characteristics at specification and design time [2]. Since the goals will drive the system at runtime, we can analyze various properties to see if they meet desired system objectives before we continue with design and implementation. In this paper, we use Petri Nets (PNs) [3] to model the runtime semantics of GMoDS specification models, which can then be used with a variety of PN tools to perform sophisticated analysis. PNs are a common graph-based approach to modeling distributed and concurrent systems. The key elements of a PN include tokens, places, arcs, and transitions. Tokens exist in places and can are consumed and created by transitions. Arcs connect places to transitions. The benefits of using PNs to model goal models lies in their ability to be analyzed using an number of existing tools [4]. Specific problems of interest related to multiagent systems include reachability, liveness, and boundedness. Reachability allows us to determine if a particular marking is reachable from the start state. Often we use this kind of analysis to determine if undesirable states are reachable. Liveness analysis looks at whether the various transitions can fire or not. If a transition is dead, then it can never fire, which often indicates a problem in the original goal mode. Boundedness looks at the number of tokens any particular place may contain. Since we model active goals as places in the PN, the boundedness of various places can give us an indication of the number of times a given goal type need be achieved, thus giving the designer insight into the importance to assign to the achievement of various goals.

2 Capturing GMoDS with Petri Nets

To model the execution semantics of GMoDS models in terms of Petri Nets, we use Colored Petri Nets (CP-nets) [3] extended with inhibitor arcs and prioritized transitions. CP-nets allow tokens to take on values (or colors) to distinguish various types of tokens. The four types of arcs used in our modeling are shown and defined in Fig. 1. The only extension to basic PN arcs that we use is the inhibitor arc, which ensures a transition *cannot* fire if there are tokens in the associated place.

We also use a two-priority transition scheme. Priority transitions are annotated with a modified transition notation as shown in Fig. 2, while regular transitions are shown as traditional transitions. All priority transitions fire before regular transitions; however, within priorities, the firing sequence



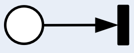
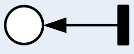
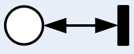
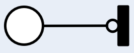
Place p	Name	Condition	Effect
	Input	M(p) > 0	M'(p) = M(p) - 1
	Output	none	M'(p) = M(p) + 1
	Reserve	M(p) > 0	M'(p) = M(p)
	Inhibitor	M(p) = 0	M'(p) = M(p)

FIG. 1. ARC TYPES. M(P) REFERS TO THE MARKING (NUMBER OF TOKENS IN) PLACE P AND A TICK MARK IS USED TO INDICATE THE MARKINGS AFTER THE TRANSITION HAS FIRED.

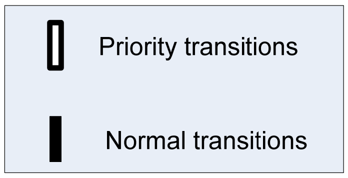


FIG. 2. TRANSITIONS NOTATIONS

is non-deterministic. Priority transitions are generally used to represent automatic instantiation or achievement of goals. For example, when both subgoals of an AND-decomposed goal are achieved, the parent goal should be achieved instantaneously. Normal transitions are generally used to model events that occur after some time has passed, such as goal achievements or triggering of new goal instances.

Colored tokens are used to capture the semantics having multiple instances of the same goal. One of the powerful features of GMoDS is the ability to instantiate new goals based on events that occur during system operation, which is called *triggering* a goal. CP-nets allow us to keep track of individual instances of goals of the same type.

Five key GMoDS elements must be translated into Petri Nets. These are leaf goals, AND-decomposed goals, OR-decomposed goals, precedence relations, and triggers relations. The interaction of precedence and triggers relation require modifications to the basic formalisms.

2.1 Leaf goals

Leaf goals are simple goals that must be achieved by agents. They are modeled as two places separated by an *achievement* transition (Fig. 3). Essentially, the first place, which we will call the *active place*, represents when the goal is in the *active* state while the transition signifies that it has been achieved. The second, or *achieved place*, represents that the goal has been achieved and can be used to trigger achievement of its parent goal (see AND- and OR-decomposed goals below). We typically decorate the active place with the name of the goal and the achievement place with the goal name and a tick mark.

2.2 AND-decomposed goals

AND-decomposed goals are goals whose sub-goals must all be achieved in order for it to be

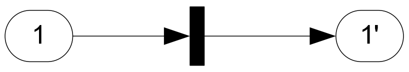


FIG. 3. LEAF GOAL FORMALISM

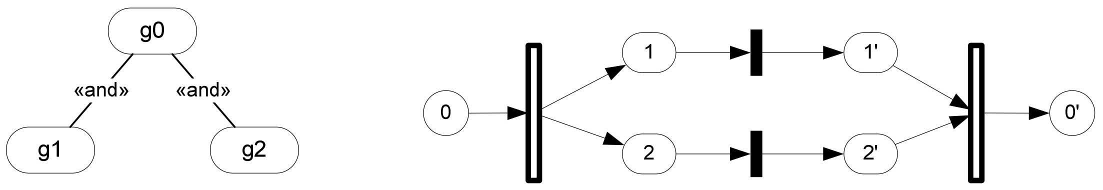


FIG. 4. AND-DECOMPOSITION FORMALISM

achieved. We formalize AND-decomposed goals as shown in Fig. 4. The parent goal is represented by two places and two priority transitions, with the definition of its sub-goals in between the transitions. There is a single input arc from the parent's active goal place to the first priority transition, the *activation* transition, with output arcs leading to the places of each of its sub-goal active places. Modeling the activation transition as a priority transition ensures that once the parent goal becomes active, each of its sub-goals will become active before any additional goal achievements can occur. Likewise, there is an input transition from each of its sub-goal's achievement place to the parent's second priority transition, the *achievement transition*, which is also modeled as a priority transition to ensure that the parent goal is achieved immediately after its last sub-goal has been achieved. The achievement transition has an output arc to the parent goal's achievement place. (If a sub-goal achievement transition occurs, no other priority transition can be enabled, thus guaranteeing immediate transition and achievement of the parent goal.)

Fig. 4 shows a simple example of an AND-decomposed goal, g0, with two sub-goals, g1 and g2. Once a transition fires placing a token on place 0, the activation transition fires (at least before any lower level goal achievement transitions) placing a token in both place 1 and place 2. The tokens will remain there until their achievement transitions fire (non-deterministically) signifying achievement of the goal. Once both transitions have fired, there will be a token in place 1' and 2' and the goal 0's achievement transition will fire placing a token in place 0' signifying achievement of the goal g0.

2.3 OR-decomposed goals

OR-decomposed goals are goals where at least one of its sub-goals must be achieved in order for it to be achieved. A key feature of the GMoDS semantics is that once one of the subgoals has been achieved, all the other subgoals are *obviated*, which is a state where they are not achieved or failed, but just no longer of use. The notion of *obviation* makes formalization a little more complicated. As shown in Fig. 5, an OR-decomposed goal is formalized by using a priority *activation* transition to activate each subgoal when the parent becomes active. Each subgoal is formalized using the basic leaf goal formalization of Fig. 3, augmented with additional places and transitions to deal with goal obviation. To ensure that parent goal is achieved immediately when a subgoal is achieved, the

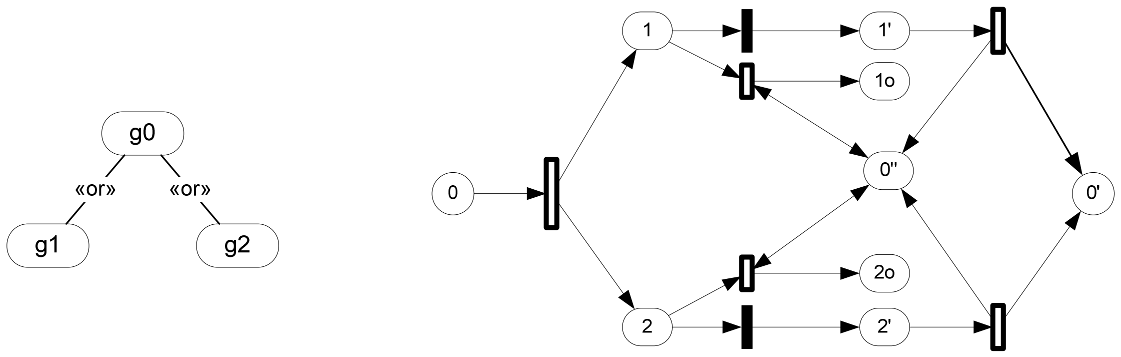


FIG. 5. OR-DECOMPOSITION

achieved place of the subgoal is tied directly to a unique achievement transition for the parent. To keep other subgoals from being achieved, a second achievement place (decorated with a double tick mark) is added to the parent goal. Next, a priority transition with an input arc from the active subgoal place and a reserve arc from the second parent achievement place is also added. An output arc leads from each priority transition to a new *obviated* place (decorated with an "o" suffix). These priority *obviation* transitions ensure that as soon as the parent goal is achieved, all tokens in the active places of the other subgoals are removed and placed into the obviated places.

Fig. 5 shows an example where the parent goal, g0, is OR-decomposed into two subgoals, g1 and g2. When g0 is activated, a token is placed in place 0, which immediately enables the activation transition and placing tokens in place 1 and 2. As soon as the (non-priority) achievement transition fires for subgoal 1 or 2, the associated achievement transition for g0 fires placing a token in place 0' and 0". Then, assuming goal g1 was achieved, g2's obviation transition fires removing the token from place 2 and placing a token in place 2o. Notice that the obviation places may be removed without affecting the semantics of the model; they are only included for clarity.

2.4 Precedence relations

If goal A *precedes* goal B, then goal A (the preceding goal) must be achieved before goal B (the preceded goal) can become active. Our formalization for this simple case of one goal preceding another is shown in Fig. 6. In this case, a second achievement place (decorated with a double tick mark) is added to preceding goal and a reserve arc is added from the second achievement place to a priority transition that is inserted before the active place of the preceded goal. A *preceding place* (labeled with a 'p' for preceded) is added with an output to the priority transition to hold token that activate the preceded goal once the preceding goal has been achieved. (A reserve arc is used in case place 2 triggers goal where we could have multiple tokens in the 2p place.)

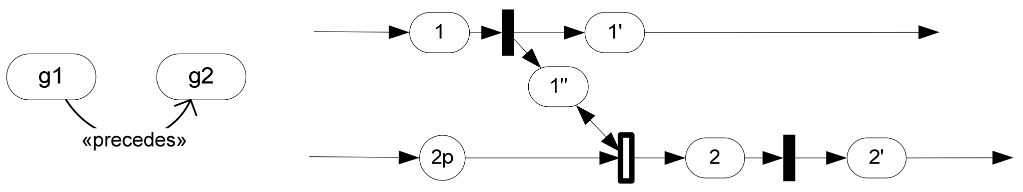


FIG. 6. SIMPLE PRECEDENCE FORMALISM

In Fig. 6, goal g1 must be achieved before goal g2 can become active. Thus, when goal g1 and g2 are instantiated, a token is placed in place 1 and place 2p. Eventually, the achievement transition for 1 will fire placing a token in both 1’ and 1” causing the priority transition to fire and place a token in place 2. Notice that 1’ will still hold a token signifying the achievement of goal 1. When the achievement transition for goal 2 fires, a token is placed in 2’. If more than one goal precedes a goal, then all preceding goals must be achieved before the preceded goal can become active. Formalization of this situation simply requires multiplication application of the formalism shown in Fig. 6. However, if a single goal precedes multiple goals, we use the same formalism with a separate reserve arc attached to each precedence transition.

2.5 Triggers relations

If a triggers relation exists between goal A (the triggering goal) and B (the triggered goal), then any time a specified event occurs while goal A is being achieved, a new instance of goal B is created. In our formalism, we represent instances of goals by colored tokens in the active place for a goal. Our formalism for representing triggers is shown in Fig. 7. A “Next” place is added with an initial marking of “1” that will allow us to create unique token colors (numbers) for each triggering. There is a transition placed between the active places of the triggering and triggered goals with a reserve arc from the output place of the triggering goal to the transition and a colored output arc from the transition to the active place of the triggered goal labeled with ‘n’. Additionally, there is an input arc from the Next place to the transition labeled with ‘n’ that specifies that the value of the token in Next will be transferred when a trigger occurs. An output arc from the transition to the Next place labeled with ‘n+1’ places a new token in Next whose value is one more than the token used to enable the transition. All arcs related to the triggered goal are labeled with ‘n’ to keep track of the value of the tokens in the net.

In Fig. 7, when a token appears in place 1, the trigger transition can fire multiple times (or none) before the achievement trigger removes the token from place 1 and places it in place 1’. Each time the trigger transition fires, a token is placed in place 2 *without* removing the token from place 1. The value of the token in Next is used as the value of the token created in place 2. The achievement

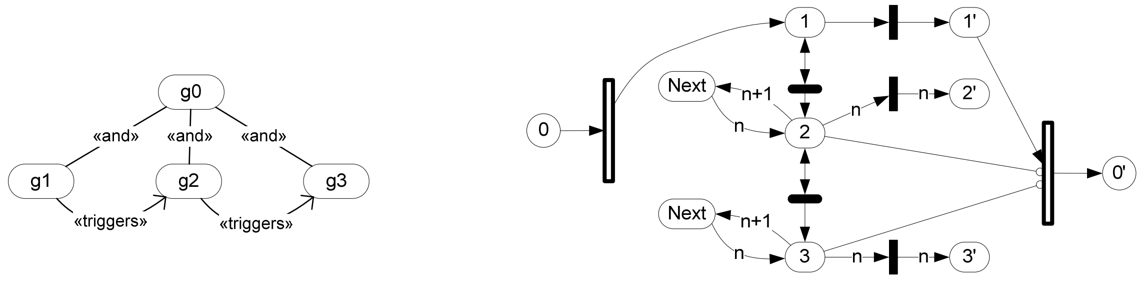


FIG. 7. AND-DECOMPOSITION FORMALISM WITH TRIGGERED SUBGOALS

transition for goal 2 can fire as long as there are tokens in place 2 (representing instances of goal 2 being active). Each time g2's achievement transition fires, a token is removed from place 2 and a token with the same color (number) is placed in place 2'.

Triggered goals require modifications to the simple definitions discussed above. Specifically, the AND-decomposition and precedence require additional machinery with triggered goals. Each of these situations will be discussed below.

2.5.1 AND-decomposition with Triggered Subgoals

An AND-decomposed parent goal requires *all* of its subgoals to be achieved before it is achieved. Thus, with triggered subgoals, as shown in Fig. 8, g1 and *all* instances of goal g2 and *all* instance of goal g3 must be achieved to achieve goal g0. The modification to the basic AND-decomposition formalism makes three changes. First, the output arcs from the activation transition for the parent to the active places of triggered goals are removed. Second, inhibitor arcs are inserted from the active place of each triggered goal to the achievement transition for the parent goal (actually, we have to have confirmation that all possibility of creating tokens in the active place of each triggered goal is no longer possible – more on that below). Finally, the input arcs from the achievement places of the triggered goals are removed.

In Fig. 8, we see an example where goal g0 is AND-decomposed into g1, g2, and g3 and that g1 triggers g2 and g2 triggers g3. In the formalization, we see that only place 1 has an input arc from the

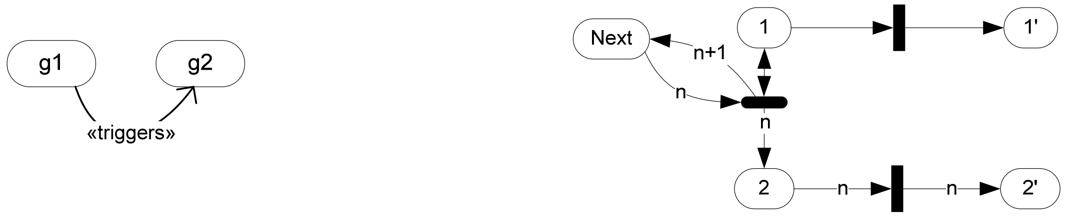


FIG. 8. TRIGGERING FORMALISM

g0’s activation transition signifying the active state of goal g0 (and thus g1). Place 2 and 3 do not have such arcs as they do not become active until triggered. We also see inhibitor arcs from place 2 and 3 to the achievement transition for goal g0. In essence, the formalism states that goal g0 is achieved if goal 1 is achieved (there is a token in place 1’), all instances of goal 2 (if any) have been achieved (there are not tokens in place 2), and all instances of goal 3 (if any) have been achieved (there are not tokens in place 3). Notice that if a token is placed in place 0, the priority transition will fire placing a token in place 1. At this point, both the achievement transition for goal 1 and the trigger transition for goal 2 are enabled. If the achievement transition fires, a token will be placed in 1’ (and removed from 1) thus causing the achievement transition to fire signifying the achievement of goal 0 by placing a token in place 0’. If the trigger transition fires, a colored token will be placed in place 2 and a regular token in place 1 (via the reserve arc) thus enabling the achievement transitions for goal g1 and g2 as well as the trigger transition for goal g3. Eventually, the achievement transition for goal g1 will fire thus eliminating the possibility that any more tokens will be created in place 2. Likewise, eventually, the achievement transition will fire for goal 2 enough time to remove all the tokens for place 2 eliminating the possibility of creating more tokens in place 3. Once place 2 and 3 are empty (with place 1’ already having a token) g0’s achievement transition fires signifying the achievement of goal 0 by placing a token in place 0’.

A more complex form of a chain of triggers is shown in Fig. 9 where a subgoal of goal g01 triggers the subgoal of goal g02. To determine when goal g02 is achieved, we must be sure all instances of goal g3 have been achieved and no more instances of g3 can be created, which will require us to trace back to ensure no more instances of g2 can be created requiring that goal g1 be achieved. Thus, the achievement of g02 requires the achievement of subgoals of g01. Our formalism for this example is shown in Fig. 9. The net is constructed using the patterns demonstrated above. However, there are two interesting aspects to this formalization. First, notice that since g3 is triggered and is the only subgoal of g02, the priority transition with an input arc from place 02 has not output arc, which is because goal g3 must be triggered. Also, notice that the achievement transition for goal g02 has

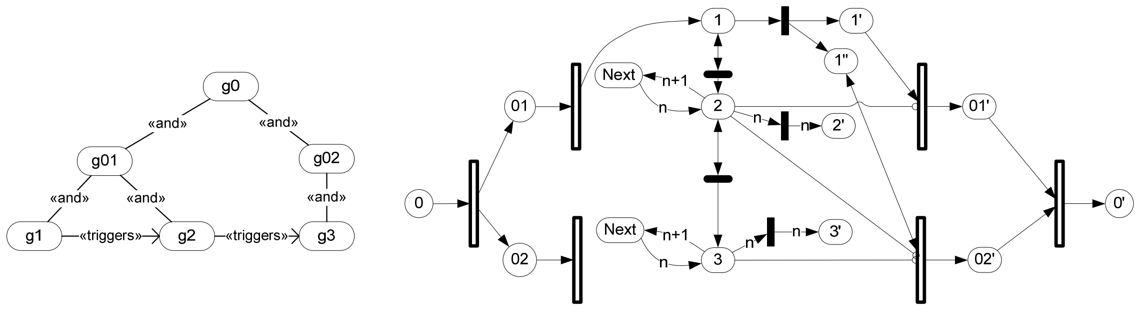


FIG. 9. COMPLEX TRIGGERS EXAMPLE

input and inhibitor arcs from the subgoal places of goal 01. This is due to the fact that to ensure no more tokens can be placed in place 3, we have to ensure that there are no more tokens in place 2 (its triggering goal’s active place) and that there can be no more tokens can be added to place 2. Generally, this requires us to trace back trough all possible triggering goals until we reach a goal that is not triggered and then verify that that non-triggered goal has been achieved. In this case, we require there be a token in place 1”, an achievement place for goal g1.

In the Petri Net example in Fig. 9 we see that when a token is inserted into place 0, the priority transition will fire placing tokens into places 01 and 02 and removing it from place 0. These will fire one after the other putting a token in place 1 and removing tokens from place 01 and 02. At this point, the example follows pretty much like that of Fig. 7. If the achievement trigger fires first removing the token from place 1 and creating tokens in places 1’ and 1”, both the achievement transitions for 01 and 02 will be enabled. When fired, new tokens will be placed in 01’ and 02’ and the achievement transition for 0 will fire placing a token in place 0’. If the trigger transitions fire between place 1 and 2 and place 2 and 3 before the achievement transition fire removing the token from place 1, the net will simulate multiple instances of goals g2 and g3 being created. After the achievement transition fires removing the token from place 1, the achievement transition will fire enough times to remove all tokens from place 2 after which, the achievement transition will fire placing a token in place 01’. At this point there may be tokens in place 3. Since no more tokens can be put into place 3, its achievement transition will eventually fire enough times to empty place 3, thus enabling the achievement transition for 02. When fired, a token will be placed in place 02’ and the achievement transition for 0 will be enabled placing a token into place 0’ signifying the achievement of goal g0.

2.5.2 Triggers and Precedence Combined

Combining triggers and precedence relations is similar to the determining when a parent goal with triggered subgoals is achieved. In fact, the formalism is almost identical. If g2 precedes g3 and g2 is a triggered goal, we must ensure that all instances of g2 have been achieved before we can move g3 to the active state. If g2 is triggered by g1, the only way to ensure that all instances of g2 have been achieved is to ensure g1 has been achieved. If g1 is *not* triggered, we simply check for the achievement of g1. An example and its formalization are shown in Fig. 10. The precedence is modeled with a priority transition with an inhibitor arc from its triggered predecessor goal and an input arc from a secondary achievement place from the non-triggered goal that triggers the preceding goal.

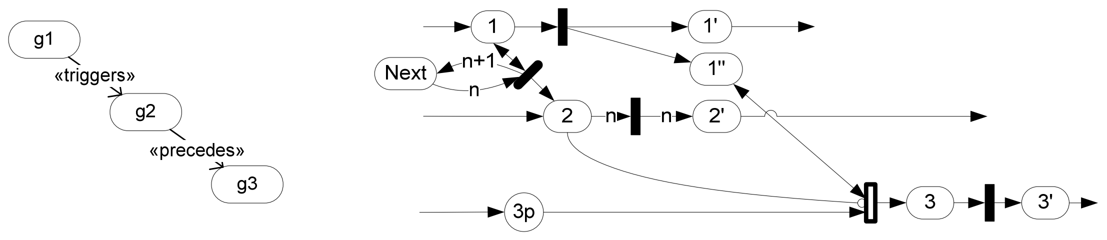


FIG. 10. PRECEDENCE WITH TRIGGERED GOAL

In Fig. 10, if we assume a token has been put in place 1 and place 3p then g1 is active, while g3 is preceded. If the achievement transition fires placing tokens in places 1' and 1", then the precedence priority transition is enable and fired placing a token in place 3 signifying that g3 is now active (no instances of g2 were created). If the trigger transition fires creating tokens in place 2 before the achievement transition is fired creating a token in place 1' and 1", the achievement transition must fire enough times to remove all the token from place 2 before the precedence transition is enabled and a token can be created in place 3.

A slightly more complicated version of a chain of triggered goals in shown in Fig. 11. This example just shows how an inhibitor arc is required from the active place of all triggered goals in the chain while an input arc is required from an achievement place of the non-triggered goal. In this example, g1 must have been achieved (a token in place 1") and all instances of g2 and g3 achieved (no tokens in place 2 or 3) before the precedence transition can be enabled.

3 Example

An example GMoDS Goal model is shown in Fig. 12. The only goal that is active when the system starts in goal 1. Divide Area. There is actually only a single event, newArea, that triggers two different goals, *2. Pickup area* and *3. Clean area*. In addition, there is a precedence relation between *2. Pickup area* and *3. Clean area*. The Clean area goal is OR-decomposed into two ways to clean the area, either Clean tile by Sweep Area and then Mop Area, or simply Vacuum Area.

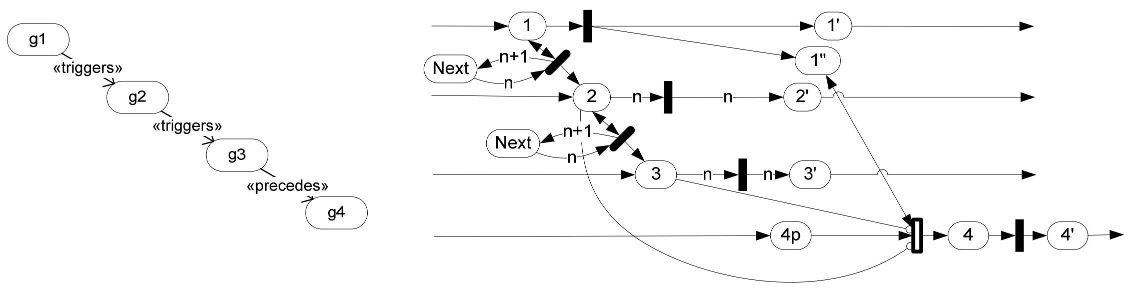


FIG. 11. PRECEDENCE WITH TRIGGERS CHAIN

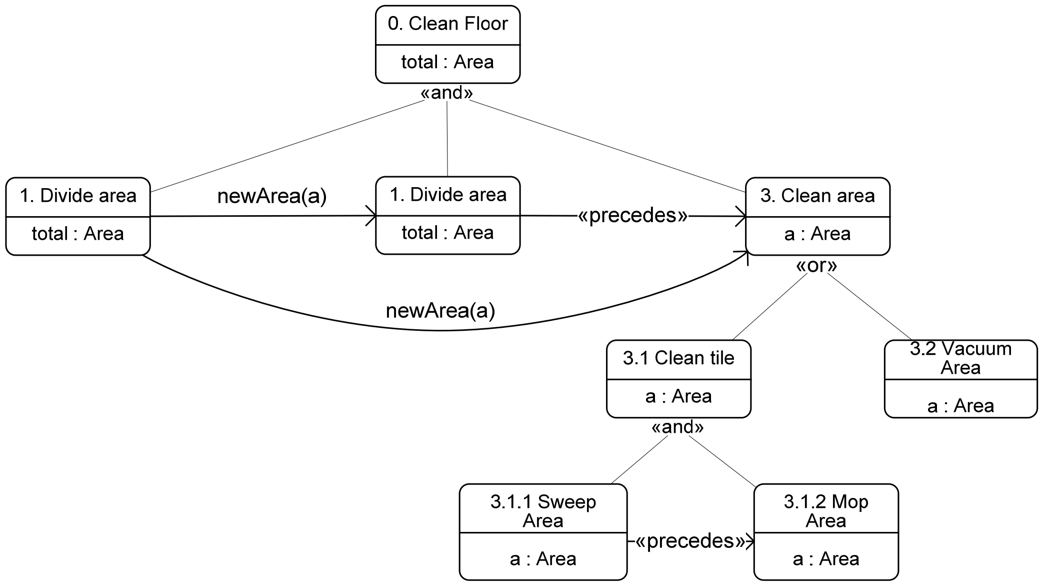


FIG. 12. CRFCC GOAL MODEL

The Petri Net formalization of Fig. 12 is shown in Fig. 13. Since the event *NewArea* triggers both goals 2 and 3, only a single triggering mechanism is included in Fig. 13; otherwise, we could trigger unrelated instances of goals 2 and 3, which is no legal given Fig. 12. Goal 3 is further decomposed using the OR- and AND-decomposition mechanism discussed above.

To see if we have achieved all instances of goal type *3. Clean Tile*, we add an extra place, 3t, that gets a copy of each colored token placed in place 3p. The tokens are removed from 3t when the achievement transitions are fired for goal 3 with those color tokens. Place 3t is connected to the g0 achievement transition via an inhibitor arc to ensure all instances of goal type 3 have been achieved

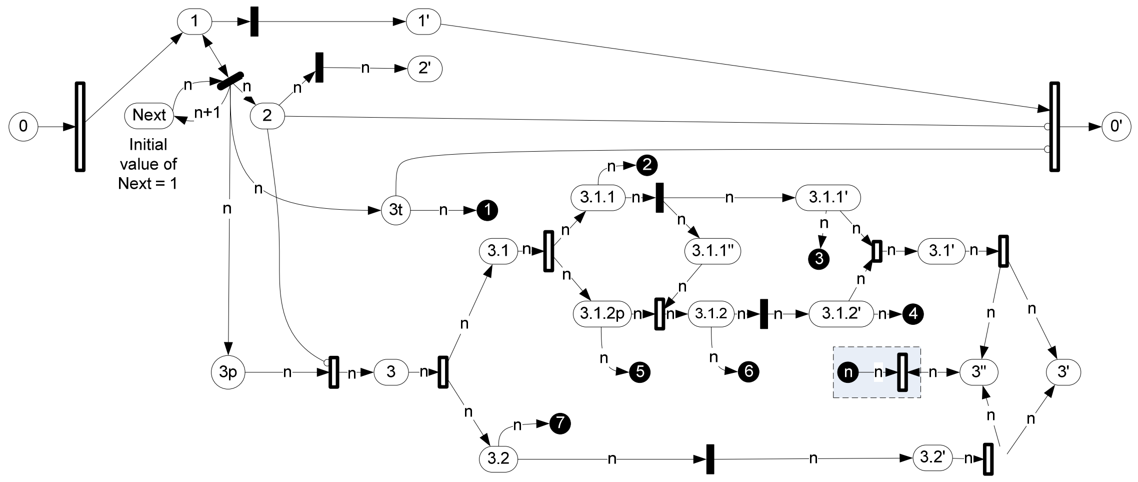


FIG. 13. EXAMPLE PETRI NET

before g0 is achieved. (Note: we use the • notation to signify that the output arcs to the • are connected to the input arc with the • and we assume the input arc and its associated transition (in grey box) are instantiated multiple times based on the number of connected output arcs.

4 Conclusions and Future Work

We implemented the PNs shown in this paper including the CRFCC example of Fig. 12 in TimeNET[1], a tool for modeling and analysis of stochastic, colored PNs. The only PN attribute not directly supported by TimeNET was inhibitor arcs for colored tokens; however, TimeNET did provide a global guard on transitions, which allowed us to specify the same information using expressions such as #P2==0, which states that for the transition to fire, the number of tokens in place P2 must be 0.

The next step of this research includes determining what kinds of analysis can be done using existing Petri Net tools and how they can be of use in engineering of GMoDS based systems.

5 References

1. Scott A. DeLoach & Matthew Miller. A Goal Model for Adaptive Complex Systems. International Journal of Computational Intelligence: Theory and Practice. Volume 5, no. 2, 2010. (in press).
2. Robby, Scott A. DeLoach, Valeriy A. Kolesnikov. Using Design Metrics for Predicting System Flexibility. in Luciano Baresi, Reiko Heckel (eds.) Fundamental Approaches to Software Engineering: 9th International Conference, FASE 2006, Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 27-28, 2006. Springer LNCS Vol 3922, pp 184-198, 2006.
3. Jensen, K. 1996 *Coloured Petri Nets (2nd Ed.): Basic Concepts, Analysis Methods and Practical Use: Volume 1*. Springer-Verlag.
4. TGI group, University of Hamburg. “Petri Net World. Tools and Software.”May 2010. http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/.

[1] http://www.tu-ilmenau.de/TimeNET