Autonomous Units to Model Games *

Hans-Jörg Kreowski, Sabine Kuske, Hauke Tönnies

University of Bremen, Department of Computer Science
P.O.Box 330440, D-28334 Bremen, Germany
{kreo, kuske, hatoe}@informatik.uni-bremen.de

Abstract: Communities of autonomous units are devices to model the interaction of independent processes in a rule-based and graphical way. In this paper, the framework is proposed to describe the interplay of players of a game following the intuition that a player is an autonomous unit playing the game according to the rules of the game but driven by its own possibilities and decisions independent of the other players.

1 Introduction

Communities of autonomous units are introduced and investigated as a formal graph-centered and rule-based framework for the modeling and analysis of interacting processes in logistics and computer science (cf. [HKHK+07], [KK07], [HKK06], [HKK09]). In this paper, we propose tentatively to use communities of autonomous units as modeling devices for games as they are studied in game theory and applied in economy and social sciences (cf. [vNM44], [Pet08], [FT91], [SLB09]). The idea is to provide a common graph-transformational approach that allows one to describe and design games in an intuitive, flexible and precise way using the following features:

(1) A game takes place in some environment (e.g. a board or some kind of "playfield") that is often suitably represented by a graph. (2) The players are individual entities acting independently of each other in an autonomous way. Usually, actions have local effects on the environment so that they can be specified by rules and performed by rule applications in a meaningful way. (3) Usually, a player can choose among several possible next actions in a given situation (otherwise there would be not much game to play). The choice is based on some kind of decision mechanism, strategy, or control condition. (4) Besides the players, there may be other dynamic entities (like referees) that are independent of the players acting according to their own rules and decisions. (5) Moreover, a game follows some kind of overall discipline (like the order in which players can act) which may be seen as a global control condition.

*The authors would like to acknowledge that their research is partially supported by the Collaborative Research Centre 637 (Autonomous Cooperating Logistic Processes: A Paradigm Shift and Its Limitations) funded by the German Research Foundation (DFG).

Summarizing, a game may be considered as a community of autonomous units where the players and other dynamic entities are modeled as units with rules and control conditions. In Section 2, the basic concepts of autonomous units are recalled. In Section 3, the modeling of games by communities of autonomous units is illustrated by the small example of *rock – paper – scissors* while the more complex example of *Ludo* is sketched in Section 4.

The hope behind our proposal to use communities of autonomous units as a common framework for the modeling of games is the following: (1) Graphs and graph transformation rules are intuitive and well-understood concepts to describe states, environments, structures, etc. and their step-wise dynamic changes. (2) Control conditions are quite generic means to provide autonomous units, like players, with decision mechanisms and strategies. (3) Graph transformation rules may be applied sequentially or in parallel so that players can act one after the other or simultaneously, whatever the game requires. (4) Graph transformation tools like AGG, GrGen and others can be used to run the game on a visual level and to observe the effects and results. (5) The formal semantics based on rule application sequences provides the possibility to prove properties of games like termination, chances to win, etc.

2 Communities of Autonomous Units

In this section, we recall the basic notions of communities of autonomous units as needed in the following. Communities of autonomous units are rule-based graph-transformational systems. Their units act and interact autonomously in a common environment, respresented as a graph. Every autonomous unit consists of a set of auxiliary units, a set of rules, a control condition, and a goal. The rules specify the potential actions and can be applied to transform the common environment. The control condition allows the unit to decide about its next action and, in this way, restrict the non-determinism of rule application. The goal serves to specify graphs that should be reached. The auxiliary units are a syntactic structuring means for deviding large rule sets and control conditions into smaller manageable pieces.

The basic ingredients are provided by an underlying graph transformation approach consisting of a class $\mathcal{G}$ of graphs, a class $\mathcal{X}$ of graph class expressions, a class $\mathcal{R}$ of rules, and a class $\mathcal{C}$ of control conditions. Every graph class expression e specifies a class of graphs $SEM(e) \subseteq \mathcal{G}$. Every multiset of rules $p \in \mathcal{R}_*$ specifies a binary relation $\underset{p}{\Longrightarrow} \subseteq \mathcal{G} \times \mathcal{G}$.[1] This means that many rules and each single rule at multiple matches can be applied in parallel. Every control condition $c \in \mathcal{C}$ specifies a set $SEM(c)$ of sequences of graphs prescribing a certain order of rule applications in this way. The notions of control conditions and graph class expressions are very generic and include logic and grammatical descriptions of graph classes and rule applications sequences.

Definition 1 (Units)**.** A *unit* is a system $u = (U, P, C, g)$ where U is a set of (imported) *units*, $P \subseteq \mathcal{R}$ is a set of rules, $C \in \mathcal{C}$ is a control condition and $g \in \mathcal{X}$ is the *goal*. The rule set of u will be also denoted by P_u. The unit u is *autonomous* if it is not imported by

[1] For a set A, A_* denotes the set of multisets over A.

another unit. All other units are called *auxiliary*. Moreover, we assume that an auxiliary unit does not have goals of its own meaning that all graphs are acceptable, i.e. $SEM(g) = \mathcal{G}$.

To avoid technical complications, we assume that the import structure of an autonomous unit can be flattened so that the import component becomes empty. With respect to the rules, this is easily done by collecting all rules involved in the import structure of an autonomous unit. With respect to control conditions, it is more complicated, but is not considered explicitly. In other words, the structuring is purely syntactical to assure small components. In this way, the semantics of communities of autonomous units can be defined for units without import.

In general, the rules of an autonomous unit aut are applied in parallel with rules of other co-existing autonomous units. To reflect this without explicit reference to the other units, the parallel semantics of aut is defined w.r.t. a set of metarules that specify actions not performed by aut. In this way, aut specifies all sequences of graphs that on one hand can be obtained via the parallel applications of the rules in the rule set of aut and the metarules, and that on the other hand are allowed by the control condition. If such a sequence meets the goal, it is called successful.

Definition 2 (Parallel semantics)**.** Let $aut = (\emptyset, P, C, g)$ be an autonomous unit, let $\mathcal{MR} \subseteq \mathcal{R}_*$. Then the *parallel unit semantics* of aut, denoted by $PAR_{\mathcal{MR}}(aut)$ consists of all sequences $s = (G_0, G_1, G_2, \cdots)$ such that for $i = 1, \cdots, |s| - 1$ if s is finite and for $i \in \mathbb{N}^+$ if s is infinite, $G_{i-1} \underset{p+m}{\Longrightarrow} G_i$ for some $p \in P_*$ and $m \in \mathcal{MR}$,[2] [3] and $s \in SEM(C)$. The sequence s is called *successful* if s is finite and $G_{|s|-1} \in SEM(g)$, or if there is an infinite monotone sequence $i_0 < i_1 < \cdots$ such that $G_{i_j} \in SEM(g)$ for all $j \in \mathbb{N}$.

A community consists of a set of autonomous units, a specification of initial graphs, a global control condition, and an overall goal.

Definition 3 (Community)**.** A *community* is a system $COM = (Init, Aut, Cond, Goal)$ where $Init \in \mathcal{X}$ is the *initial environment specification*, Aut is a set of autonomous units, $Cond \in \mathcal{C}$ is the *global control condition* and $Goal \in \mathcal{X}$ is the global *goal*.

The parallel semantics of a community is composed of all graph sequences s so that s is in the parallel semantics of every autonomous unit in the community, s is allowed by the global control condition and s starts with an initial environment. The sequence s is called *successful* if it reaches the overall goal.

Definition 4 (Parallel community semantics)**.** Let $COM = (Init, Aut, Cond, Goal)$ be a community. The *parallel community semantics* of COM, denoted by $PAR(COM)$, consists of all sequences $s = (G_0, G_1, \ldots)$ such that $G_0 \in SEM(Init)$, $s \in SEM(Cond)$,

[2] $|s|$ denotes the number of graphs in s.

[3] For two multisets a and b, the multiset $a + b$ is obtained by summing up the multiplicity of all elements in a and b.

and $s \in PAR_{\mathcal{MR}(aut)}(aut)$ for all $aut \in Aut$ where

$$\mathcal{MR}(aut) = (\bigcup_{aut' \in Aut \setminus \{aut\}} P_{aut'})_*.$$

Successful graph sequences are defined as in Definition 2.

It may be noted that the semantics of the community and the semantics of one of its units *aut* coincide if the latter is restricted to initial graphs and constructed with $\mathcal{MR}(aut)$ as metarules. Moreover, the semantics of aut must be intersected with the sequences allowed by the global control condition and by the control conditions of the other units in the community.

For modeling sequential transformation processes, communities of autonomous units are equipped with a sequential semantics which is a special case of the parallel semantics. In this case the metarules are rules instead of multisets of rules, and the sequential semantics of an autonomous unit aut consists of all graph sequences where in every step either a rule or a metarule is applied. As in the parallel case, these sequences must obey the control condition of aut. Moreover, the sequential community semantics is given by all sequences s such that the first graph in s is an initial environment, s is allowed by the control condition of the community and all its autonomous units, and in every step exactly one autonomous unit of the community is active by applying exactly one of the rules in its complete rule set.

Examples of the introduced concepts can be seen in the following.

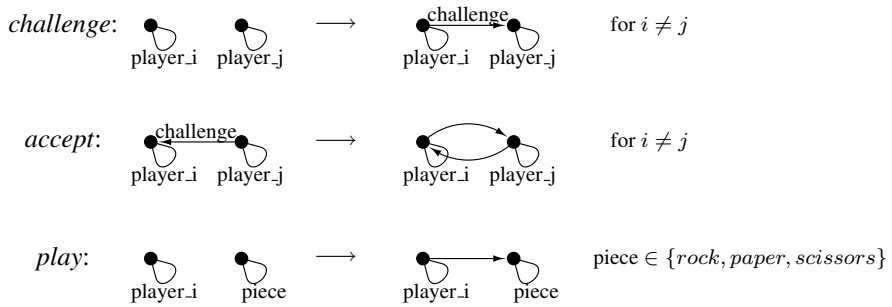
3 Modeling *rock – paper – scissors* with Autonomous Units

The aim of this section is to model the well-known game *rock – paper – scissors* with autonomous units. To diversify the example, an arbitrary number of players is allowed. Every participant of the game gets a number and is associated with the autonomous unit depicted in Fig. 1. Its actions, modeled by graph transformation rules, are the following. To play against somebody, the player has to *challenge* another player in the environment, who, in turn, can *accept* the challenge (or not). Technically, the agreement is realized by inserting two edges between the two player in reverse direction. The rule *play* finally models the action to take either the *rock*, the *paper* or the *scissors* by drawing an edge between the player node and the node representing the chosen token. The control condition assures that an arbitrary number of challenges is offered in parallel (indicated by the exponent $||$) before an arbitrary number of challenges is accepted and, finally, a single play rule is applied.

The autonomous unit *referee*, depicted in Fig. 2, is responsible for determining the winners. For every combination out of the nine possible ones, a rule exists establishing either a winner, in case the two player chose different tokens, or a draw, if both players chose the same token. Two rule patterns covering all the possibilities by using variables are shown in Fig. 2. For every playing pair a corresponding rule is applied and the winner (if there is

$player_i$

rules:

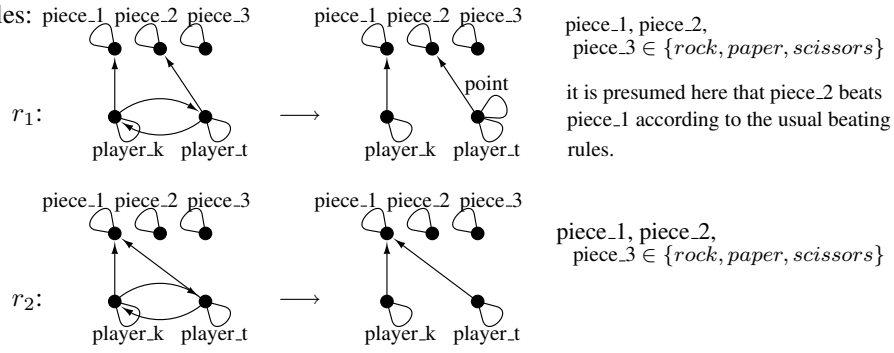


cond: $challenge^{||}; accept^{||}; play$

Figure 1: The autonomous unit $player_i$

$referee$

rules:



cond: max

Figure 2: The autonomous unit *referee*

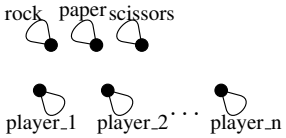


Figure 3: The initial graph

one) gets one point (modeled as a loop labeled with *point*). Due to the control condition *max*, a maximum number of rules are applied in parallel. Then the autonomous unit *clean*, which is not shown here for reasons of space limitations, deletes all unnecessary edges so that the resulting graph resembles the initial graph in Fig. 3 (plus the point loops at the player nodes). Now a new round can be played.

In total, the game can be modeled as the community $RPS = (Init, Aut, Cond, Goal)$ with the following components.

1. *Init* consists of the graph having three nodes with loops labeled with *rock, paper* and *scissors* respectively and an arbitrary, but fixed number *n* of player nodes (see Fig. 3).
2. *Aut* consists of the autonomous units $player_1, \ldots, player_n, referee, clean$.
3. *Cond* consists of the control condition $(player_1 || \ldots || player_n; referee; clean)^*$ meaning that first all the player units act in parallel, then the referee unit establishes the winners and afterwards the cleaning unit cleans up the environment. This procedure in this specific order can be repeated an arbitrary number of times (and is considered successful, if the goal is achieved).
4. The *Goal* is a natural number k that specifies all graphs that contain the initial graph and additionally an arbitrary number of point loops so that at least one player has k or more of them.

4 Modeling *Ludo* with Autonomous Units

The game *Ludo* is a more complex example, where the semantics is sequential. Due to the complexity, a presentation of an autonomous community modeling *Ludo* needs a lot more space than provided here, so just a few highlights are sketched. A more complete modeling can be found in [HKK09].

Like before, every player of *Ludo* is modeled by one autonomous unit whose rules reflect the possible actions of the player. Each autonomous unit is equipped with the goal to have all of its tokens in its home base. One further autonomous unit models the die. The

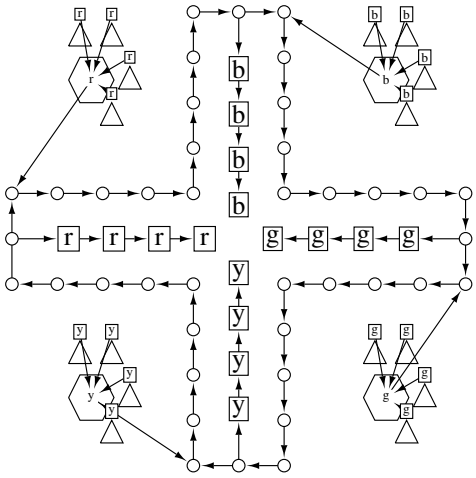


Figure 4: The initial board

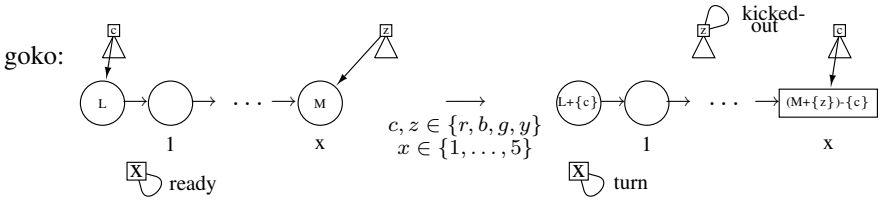


Figure 5: The rule *move and kick out*

environment consists of the graph representing the actual state of the game board. The initial graph is shown in Fig. 4 leaving out the details.

As a representative example, one rule of the unit player is shown in Fig. 5. It models the moving of one token and kicking out of a token from a different player. The rectangle node with the x represents the die and the x the thrown number. The round nodes represent the fields and the triangle node the token of a player. Note that the two triangle nodes are labeled differently, so they belong to different players. Since it is not allowed to have two tokens of the same color at one field or to enter the home base of a different player, every field token is labeled with a set of labels specifying the tokens allowed to reside. It should be noted here, that this rule is applied only if one has not thrown a six, since a six implies to play again and thus requires a different rule.

The control conditions of the player units are a good way to establish different game strategies, like keeping every token as far as possible from the opponent tokens or kicking out enemies whenever possible although putting oneself in danger. We have designed a variety of strategies mostly by means of priorities among the rules. Having no control conditions, one of the tokens that are available for moving are non-deterministically chosen, which means that the winner is completely determined by luck.

5 Conclusion

In this paper, we have proposed to apply the framework of communities of autonomous units for the modeling of games. The original intention of autonomous units has been the modeling of interacting processes in logistics and computer science. We believe that players in a game play roles quite similar to interacting processes so that it is meaningful to make the attempt and to model games as communities. To shed more light on the significance of this proposal, further research is needed: (1) More examples of games must be designed as communities of autonomous units to illustrate that the approach is intuitive and adequate. (2) In particular, more and much more sophisticated examples are needed to show that the control conditions of each unit allows to model the tactics and strategies of players in an appropiate way. (3) Moreover, it would be interesting to prove properties of games based on the description as communities and their semantics. We hope that communities of autonomous units will turn out to provide a suitable common framework for the modeling and analysis of games with the perspective of tool support for the visualization, animation, simulation and verification of games.

References

[FT91] Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, 1991.

[HKHK+07] K. Hölscher, R. Klempien-Hinrichs, P. Knirsch, H.-J. Kreowski, and S. Kuske. Autonomous Units: Basic Concepts and Semantic Foundation. In M. Hülsmann and K. Windt, editors, *Understanding Autonomous Cooperation and Control in Logistics – The Impact on Management, Information and Communication and Material Flow*, pages 103–120, Berlin Heidelberg New York, USA, 2007. Springer.

[HKK06] Karsten Hölscher, Hans-Jörg Kreowski, and Sabine Kuske. Autonomous Units and Their Semantics — the Sequential Case. In Andrea Corradini, Hartmut Ehrig, Ugo Montanari, Leila Ribeiro, and Grzegorz Rosenberg, editors, *Proc. International Conference of Graph Transformation*, volume 4178 of *Lecture Notes in Computer Science*, pages 245–259, 2006.

[HKK09] Karsten Hölscher, Hans-Jörg Kreowski, and Sabine Kuske. Autonomous Units to Model Interacting Sequential and Parallel Processes. *Fundamenta Informaticae*, 93(3):233–257, 2009.

[KK07] Hans-Jörg Kreowski and Sabine Kuske. Autonomous Units and Their Semantics - The Parallel Case. In J.L. Fiadeiro and P.Y. Schobbens, editors, *Recent Trends in Algebraic Development Techniques, 18th International Workshop, WADT 2006*, volume 4408 of *Lecture Notes in Computer Science*, pages 56–73, 2007.

[Pet08] Hans Peters. *Game Theory - A multi-level approach*. Springer Verlag, 2008.

[SLB09] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[vNM44] John von Neumann and Oscar Morgenstern. *Theory of games and economic behaviour*. Princeton University Press, 1944.