AGENTS THAT LEARN TO BEHAVE IN MULTI-AGENT SIMULATIONS

Jacinto Dávila[1] and Mayerlin Uzcátegui[1,2]
[1] CeSiMo. Facultad de Ingeniería
[2] SUMA. Facultad de Ciencias
Universidad de Los Andes
Mérida, 5101. Venezuela
email: {jacinto,maye}@ula.ve

ABSTRACT

This paper illustrates the use of Inductive Logic Programming to program agents that learn rules of behaviour from simulated histories of their embedding systems. We have shown how a ILP system can be used to learn rules in a representation very close to the one used to guide the simulation of a multi-agent system. This establishes the feasibility of embedding (resource-bounded) learners as agents that take part in simulating a complex system.

KEYWORDS

ILP, Multi-agent simulation, logic-based agents

1 Introduction.

This paper illustrates the use of Inductive Logic Programming[13, 10] to program agents that learn rules of behaviour from simulated histories of their embedding systems. We are building upon previous work in which we sketched a multi-agent theory for simulation [4, 16], which is being used as the specification of an actual simulation platform, GALATEA [5, 16]. GALATEA is being used to model and simulate the complex dynamics (social, economics and ecological processes) of a forest reserve in Venezuela [1, 14, 8].

The paper is organized as follows: Section 3 briefly shows a computational, multi-agent model of that forest reserve (explained at length in [14]) and explains the context in which the learning agent will function. In Section 4, we describe a simplified learning experiment on that multi-agent context, explaining some of the preliminar results in section 5. Section 6 concludes. However, before all that, we start in section 2 with a briefing of the agent theory on which we are working.

2 What is an agent

We, preliminary, describe an agent as a 7-tuple, state-machine:

$$< P_a, K_a, G_a, Perception_a, Update_a, Planning_a, Learning_a > \quad (1)$$

where P_a and $Perception_a$ are the percepts domain and the perception function which we will not explain for the sake of space. The set K_a and G_a are, respectively, the agent's knowledge base and the agent's goals. $Update_a$ add observations into K_a, preserving them as a collection of logical formulae. Similarly, $Planning_a$, derives new goals and influences (actions descriptions), taking into account the previous goals and the knowledge base. Finally, $Learning_a$ produces new, possibly partially refined, rules to be added to K_a.

2.1 The behaviour of an agent as a mathematical function

The behaviour of that state-machine can be compactly described as follows:

$$< k'_a, g'_a, \gamma'_a > \; = \; Behaviour_a(t, r_a, k_a, g_a, \gamma) \quad (2)$$

$$k'_a \; = \; Learning_a(Update_a(t, Perception_a(\gamma), k_a), \ldots) \quad (3)$$

$$< \gamma'_a, g'_a > \; = \; Planning_a(t, r_a, k'_a, g_a) \quad (4)$$

where t is the current simulation time, r_a is a bound for the time allocated to reasoning in the agent, $k_a \in K_a$, $g_a \in G_a$, γ is the history of actions, this far, and γ_a is the set of actions this agent will try to execute. Notice that the $Learning$ function is not specified here for the sake of space.

2.2 A multi-agent system

On that brief description of an agent and an $Transition$ function modelling the global behaviour of the simulator, we can build the mathematical description of a system populated by many of such agents. We only need to connect $Transition$ with the $Behaviour$ function for each agent, as follows:

$$< \sigma', \gamma' >= Transition(t, \sigma, \gamma \cup_a \gamma_a) \quad (5)$$

and

$$< s'_a, \gamma_a >= Behaviour_a(t, r_a, k_a, g_a, \gamma) \quad (6)$$

where s'_a is an abbreviation of $< k'_a, g'_a >$, the knowledge base and goals of agent a, t and γ as before and σ represents the state variables of the system as a whole. This links the influences from the agents' behaviour to the reaction of the environment and completes the definition of the multi-agent system.

3 An example of a MAS model

What follows is the basic layout of a model of a multi-agent system coupled with a natural dynamics.

3.1 Agent Modelling of a Forest Reserve

The model here described is an outcome of the project Biocomplexity: Integrating Models of Natural and Human Dynamics in Forest Landscapes Across Scales and Cultures [12]. It aims to model and simulate land use and changes in vegetation cover as a consequence of human actions.

As it has been explained in [1], we have being devising a collection of toy models to cater for 1) the human dynamics, using the set of conceptual tools and data structures provided by GALATEA and 2) the environmental dynamics, by integrating a cellular automaton from the SpaSim [9, 7] library into the actual simulator of a forest reserve. The data structures of GALATEA provide for the representation of the agents' goals, beliefs and observations, and, also, for a very elementary reasoning engine to deduce actions for each agent, according to its circumstances.

The original model[1] considers several instances of "settler" agents and a lumber "concessionary" agent. For the sake of space, we will only consider here a simplified version of one behavior of the settler agents in the Forest Reserve. The settlers are people of limited economical resources that arrive at the reserve aiming to improve their economical status. Initially they dedicated themselves to subsistence agriculture: they just try to maximize the benefits from their occupation of the area, without considering soil exhaustion due to poor management practices, and without much regard for ecological damage. After five years aproximately, the land loses its fertility, and the settler must move to another available place (i.e. an area not under government supervision) or expand his farm by deforesting some adjacent area. In that context, cattle-raising is always the best strategy to success, provided that they secure resources to start with. This can be done by selling the lands they have occupied. The agent rules of behaviour can be formalized as shown in figure 1. This is the normal layout of a simulation model in GALATEA's ancestor language GLIDER[15, 6], but here enriched with a logic-based description for each agent[3, 2].

```
NETWORK
 LANDSCAPE (A) ::
 // Environment model. Not shown

AGENTS
  Settler (AGENT) ::
  GOALS
    if not(occupied_land, now),
       not(supervised, now),
       abandoned_land(now)
    then be_successful(self, now+5).
    if land_does_not_produce(now),
       not(occupied_land_next(now)),
    then expand(self, now).

  BELIEFS
    to be_successful(A,T) do
        settle(A,T1),
        T1=<T, plant(A,T).
    to be_successful(A,T) do
        settle(A,T1),
        plant(A,T2),
        sale(A,T3),
        cattle(A,T).
    to be_successful(A,T) do
        settle(A,T1),
        plant(A,T2),
        cattle(A,T).

INTERFACE
// Here goes code to explain
// the effects of the agents'
// actions on the environment.
// Not shown.

INIT
// Here the initiation services.
  time_step := 10;
  ACT(LANDSCAPE, 0);

DECL
//  The data structures.
// Not shown.

END.
```

Figure 1. Partial view of the Caparo MAS Model in GALATEA

4 Agents that learns how to behave

Figure 2 shows a fragment of the code provided to the learning system PROGOL[11] as background knowledge.

Notice that it corresponds to a simplified, partial history with the actions of 10 agents. Each action is described by a predicate-like entry with its name that also annotates the performing agent and the time of execution, just like the ones in the rules in figure 1.

5 Results

Depending upon the set of learning examples (see figure 3[1]) and computing resources, PROGOL starts by producing the the following outputs:

```
[Generalising successful(ag8,5).]
[Most specific clause is]


successful(A,B) :- B=<B,
    settle(A,C),
    plant(A,D), expand(A,E),
    sale(A,E), cattle(A,B),
    cattle(A,E), cattle(A,F),
    C=<B, C=<C, C=<D, C=<E,
    C=<F, D=<B, D=<D, D=<E,
    D=<F, E=<B, E=<E, E=<F,
    F=<B, F=<F.
```

which says that *an agent A is successful at time B if she settles down, plants, expands, sales and do cattle-raising in the order indicated by the =< conditions*. This shows how the system is able to generalize from examples to rules, that will then be checked to see if they cover (positive and not negative) teaching examples. The final outcome of the learning process looks like:

```
...
[C:-9999,3,10000,0 successful(A,B)
    :- cattle(A,C),cattle(A,D).]
[C:-9999,3,10000,0 successful(A,B)
    :- cattle(A,C), cattle(A,D).]
[60 explored search nodes]
f=1,p=3,n=0,h=0
[Result of search is]

successful(A,B) :- sale(A,C), C=<B.

[3 redundant clauses retracted]
successful(A,B) :-
    sale(A,C), C=<B.
[Total number of clauses = 1]

[Time taken 0.060s]
```

which basically says that *Agent A is successful at B if she sales at C and C is before or at B.* Notice that this rule is produced after exploring many alternatives (60 nodes, or alternative rules, in this case) and choosing the one with the best evaluation score (not explained here for the lack of space, but related to the numbers accompanying the rules above).

[1]:- b stands for not b, to annotate the negative examples

```
% ag1                     % ag2
settle(ag1, 0).           settle(ag2, 0).
plant(ag1, 0).            expand(ag2, 0).
expand(ag1, 1).           expand(ag2, 1).
plant(ag1, 1).            plant(ag2, 2).
expand(ag1, 2).           plant(ag2, 3).
plant(ag1, 2).            expand(ag2, 4).
expand(ag1, 3).           plant(ag2, 4).
                          plant(ag2, 5).

% ag3                     % ag4
settle(ag3, 0).           settle(ag4, 1).
expand(ag3, 1).           expand(ag4, 1).
plant(ag3, 1).            plant(ag4, 1).
expand(ag3, 2).           expand(ag4, 2).
plant(ag3, 2).            plant(ag4, 2).
expand(ag3, 3).           expand(ag4, 3).
plant(ag3, 3).            plant(ag4, 3).
plant(ag3, 4).            plant(ag4, 4).
plant(ag3, 5).            plant(ag4, 5).

% ag5
settle(ag5, 0).           % ag6
expand(ag5, 1).           settle(ag6, 0).
plant(ag5, 1).            plant(ag6, 1).
expand(ag5, 2).           expand(ag6, 3).
plant(ag5, 2).            sale(ag6, 3).
expand(ag5, 3).           cattle(ag6, 3).
cattle(ag5, 3).           cattle(ag6, 4).
cattle(ag5, 4).           cattle(ag6, 5).
cattle(ag5, 5).

% ag7                     % ag8
settle(ag7, 0).           settle(ag8, 0).
plant(ag7, 1).            plant(ag8, 1).
expand(ag7, 3).           expand(ag8, 3).
sale(ag7, 3).             sale(ag8, 3).
cattle(ag7, 3).           cattle(ag8, 3).
cattle(ag7, 4).           cattle(ag8, 4).
cattle(ag7, 5).           cattle(ag8, 5).

% ag9                     % ag0
settle(ag9, 0).           settle(ag0, 0).
plant(ag9, 1).            plant(ag0, 1).
expand(ag9, 3).           expand(ag0, 3).
buy(ag9, 3).              cattle(ag0, 3).
cattle(ag9, 3).           cattle(ag0, 4).
cattle(ag9, 4).           cattle(ag0, 5).
cattle(ag9, 5).           sale(ag0, 6).
```

Figure 2. A simplified history in a MAS: Agents acting in a Forest Reserve. Which one is more successful?

```
   successful(ag8,5).
   successful(ag8,6).
   successful(ag7,5).
   successful(ag7,6).
   successful(ag6,5).
   successful(ag6,6).

   :- successful(ag0,5).
   :- successful(ag9,5).
   :- successful(ag1,5).
   :- successful(ag2,5).
   :- successful(ag3,5).
   :- successful(ag4,5).
   :- successful(ag5,5).
```

Figure 3. Some examples to teach the agents

With a bigger training set (using 9 positive examples), the same outcome is produced. A much bigger space is explored and the systems hits some predefined limits:

```
Resource limit exceeded
[10000 explored search nodes]
f=7,p=9,n=0,h=0
[Result of search is]


successful(A,B) :- sale(A,C), C=<B.


[9 redundant clauses retracted]
successful(A,B) :- sale(A,C),
C=<B. [Total number of clauses = 1]

[Time taken 12.190s]
```

The effect of negative examples is also meaningful, as shown by the following different output from an experiment with the same 9 positive examples, and the 7 negative examples in figure 3:

```
[7421 explored search nodes]
f=7,p=9,n=0,h=0
[Result of search is]

successful(A,B) :-
    sale(A,C), cattle(A,C).

[9 redundant clauses retracted]
successful(A,B) :-
    sale(A,C),
    cattle(A,C).

[Total number of clauses = 1]

[Time taken 5.100s]
```

Notice that, in all cases, the learner does not produce a whole plan (a complete program) but only points to crucial actions or conditions. This is due to PROGOL learning strategy of maximal compression[11] and it is certainly not a constraint if one builds in the learner within the simulation system, as we intend to do. On the contrary, it could be an useful strategy for learners that have to face an enormous store of information in the (simulated) history of the system.

6 Conclusions

We have shown how a ILP system can be used to learn rules in a representation very close to the one used to simulate a multi-agent system. This establishes the feasibility of embedding (resource-bounded) learners as agents that take part in simulating a complex system.

Acknowledgements

This work has been partially funded by CDCHT-University of Los Andes projects I-666-99-02-E and Fonacit S1-2000000819.

References

[1] M. Ablan, J. Dávila, N. Moreno, R. Quintero, and M. Uzcátegui. Agent modeling of the caparo forest reserve. In *EUROSIS 2003*, pages 367–372, Naples, Italy, October 2003.

[2] Jacinto Dávila. Actilog: An agent activation language. In *PADL2003*, LNCS, New Orleans, USA, 2003.

[3] Jacinto A. Dávila. Openlog: A logic programming language based on abduction. In *PPDP'99*, Lecture Notes in Computer Science. 1702, Paris, France, 1999. Springer. Available from: `http://citeseer.nj.nec.com/64163.html`.

[4] Jacinto A. Dávila and Kay A. Tucci. Towards a logic-based, multi-agent simulation theory. In *International Conference on Modelling, Simulation and Neural Networks [MSNN-2000]*, pages 199–215, Mérida, Venezuela, October, 22-24 2000. AMSE & ULA. Available from: `http://citeseer.nj.nec.com/451592.html`.

[5] Jacinto A. Dávila and Mayerlin Uzcátegui. Galatea: A multi-agent simulation platform. In *International Conference on Modelling, Simulation and Neural Networks [MSNN-2000]*, pages 217–233, Mérida, Venezuela, October, 22-24 2000. AMSE & ULA. Available from: `http://citeseer.nj.nec.com/451467.html`.

[6] GLIDER Development Group. *GLIDER Reference Manual, Versión 5.0*. Cesimo & IEAC, Universidad de Los Andes, Mérida, Venezuela, 2000. CESIMO

IT-02-00. Available from: `http://afrodita.faces.ula.ve/Glider/`.

[7] N. Moreno, M. Ablan, and G. Tonella. Spasim: A software to simulate cellular automata models. In *IEMSs 2002, First Biennial Meeting of the International Environmental Modeling and Software Society*, Lugano, Switzerland, 2002. Available from: `http://mistoy.ing.ula.ve/INVESTIGACION/PROYECTOS/SpaSim/SpaSim/`.

[8] Niandry Moreno, Raquel Quintero, Magdiel Ablan, Rodrigo Barros, Jacinto Dávila, Hirma Ramírez, Giorgio Tonella, and Miguel F. . Acevedo. Biocomplexity of deforestation in the caparo tropical forest reserve in venezuela: an integrated multi-agent and cellular automata model. *Environmental Modelling and Software*. to be published.

[9] Niandry L. Moreno. Diseño e implementación de una estructura, para el soporte de simulación espacial en GLIDER. Master's thesis, Maestría en Computación, Universidad de Los Andes. Mérida. Venezuela, 2001. Tutor: Ablan, Magdiel.

[10] Stephen Muggleton. Inverse entailment and progol. Technical report, The University of York, York, UK, 2002.

[11] Stephen Muggleton and Jhon Firth. *Cprogol4.4: A tutorial introduction*. Department of Computer Sciences, The University of York, United Kingdom, 2002.

[12] NSF. Biocomplexity: Integrating models of natural and human dynamics in forest landscapes across scales and cultures. http://www.geog.unt.edu/biocomplexity, 2002.

[13] David Page. Ilp: Just do it. *Lectures Notes in Artificial Intelligence*, (1866):3–18, 2000.

[14] R. Quintero, R. Barros, J. Dávila, N. Moreno, Tonella G., and M. Ablan. A model of the biocomplexity of deforestation in tropical forest: Caparo case study. In Schmidt S. Pahl, C. and T. Jakeman, editors, *iEMSs 2004*, Osnabrueck, Germany, June 2004. http://www.iemss.org/iemss2004/proceedings.

[15] G. Tonella, M. Acevedo, M. Ablan, C. Domingo, H. Hoeger, and C. Sananes. The use of glider as a tool for the simulation of ecological systems. In M.H. Hamza, editor, *Proceedings of the IASTED International Conference*, number ISBN 0-88986-196-X, pages 463–367, Anaheim, California, October 1995. Acta Press.

[16] Mayerlin Y. Uzcátegui. Diseño de la plataforma de simulación de sistemas multi-agentes galatea. Master's thesis, Maestría en Computación, Universidad de Los Andes. Mérida. Venezuela, 2002. Tutor: Dávila, Jacinto.