

Multi Agent Systems

Intelligent Agent Architectures

The Naming Game

The Naming Game Tweaked

B. Tenbergen (btenberg@uos.de)

1. Overview

- 1. Overview
- **2. Architectures of Intelligent Agents – A Sketch**
 - **2.1 How can Agent \wedge Environment interaction be formalized?**
 - **2.2 How can we make an Agent perceive?**
- 3. The Naming Game by Luc Steel – an example of a MAS
 - 3.1 What is the Naming Game?
 - 3.2 The Naming Game Formalisms
 - 3.3 How does the Naming Game work?
- 4. The Naming Game tweaked – a Research on MAS
 - 4.1 Intro and Method
 - 4.2 Results and Discussion

2. Architectures of Intelligent Agents

2.1 How can Agent–Environment Interaction be formalized?

'An agent architecture is essentially a map of the internals of an agent – its data structures, the operations that may be performed on these data structures, and the control flow between these data structures.'
(M. Wooldridge)

- Until now, the concept of an Agent was kind of abstract to us.
- Yet, before we can have a look at some nice architecture examples, we need to know how an Agent (re)acts with the environment.

2. Architectures of Intelligent Agents

2.1 How can Agent–Environment Interaction be formalized?

- The Environment, an Agent is situated in, can be described by a finite set of states:

$$\mathcal{E} = \{e, e', e'', \dots\}$$

- This set is discrete. However, a continuous environment can be represented by an augmented discrete set.
- Now, every Agent is capable of some distinct actions. Thus,

$$A_c = \{a, a', a'', \dots\}$$

is the finite set of actions.

2. Architectures of Intelligent Agents

2.1 How can
Agent–Environment Interaction
be formalized?

- So, we formulate an algorithm:

<snip>

start in environment state e ;

REPEAT

let the agent choose an action according to $e \rightarrow a$;

the environment changes to state e' because of a : $a \rightarrow e'$

UNTIL infinity ∞ or final state reached

</snip>

- Please note, that the environment can respond with a bunch of states, from which the agent has no idea.

2. Architectures of Intelligent Agents

2.1 How can Agent–Environment Interaction be formalized?

- That algorithm is called a run, denoted by r .

This run can be written as a sequence of interleaved environment states and corresponding actions:

$$r : e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \xrightarrow{\alpha_3} \dots \xrightarrow{\alpha_{u-1}} e_u.$$

- Now, we denote R as the set of those runs, with
 - R_{ac} is the subset that ends with an action and
 - R_e is the subset ending with an environment state.

2. Architectures of Intelligent Agents

2.1 How can
Agent–Environment Interaction
be formalized?

- The function that maps an action onto an environment effect is (after Fagon et al., 1995):

$$\tau : \mathcal{R}^{Ac} \rightarrow \wp(E).$$

We will assume, that every run eventually terminates.

- Finally, we need a function, that maps runs to actions:

$$Ag : R_e \rightarrow Ac$$

2. Architectures of Intelligent Agents

2.1 How can Agent–Environment Interaction be formalized?

- Wow, kind of many functions and notations... but, there aren't many to come.
- What is important to say is that environments are dertermined by their history. This means, that the preceeding sequence of states determines the successor states.
- In addition to that, the result of a specific action in a specific state is not determined, since the state transformer function allows for non-determinism.

2. Architectures of Intelligent Agents

2.1 How can
Agent–Environment Interaction
be formalized?

- We have an environment ***Env*** that is a 3-tuple:

$$Env = \langle \mathcal{E}, e_0, T \rangle$$

with \mathcal{E} being a set of states,
 e_0 being the initial state and
 T being the state transformer function.

- We then can say, that two Agents ***Ag₁*** and ***Ag₂*** are '***behaviourally equivalent*** (with respect to that environment), **iff**

$$R(Ag_1, Env) = R(Ag_2, Env)$$

2. Architectures of Intelligent Agents

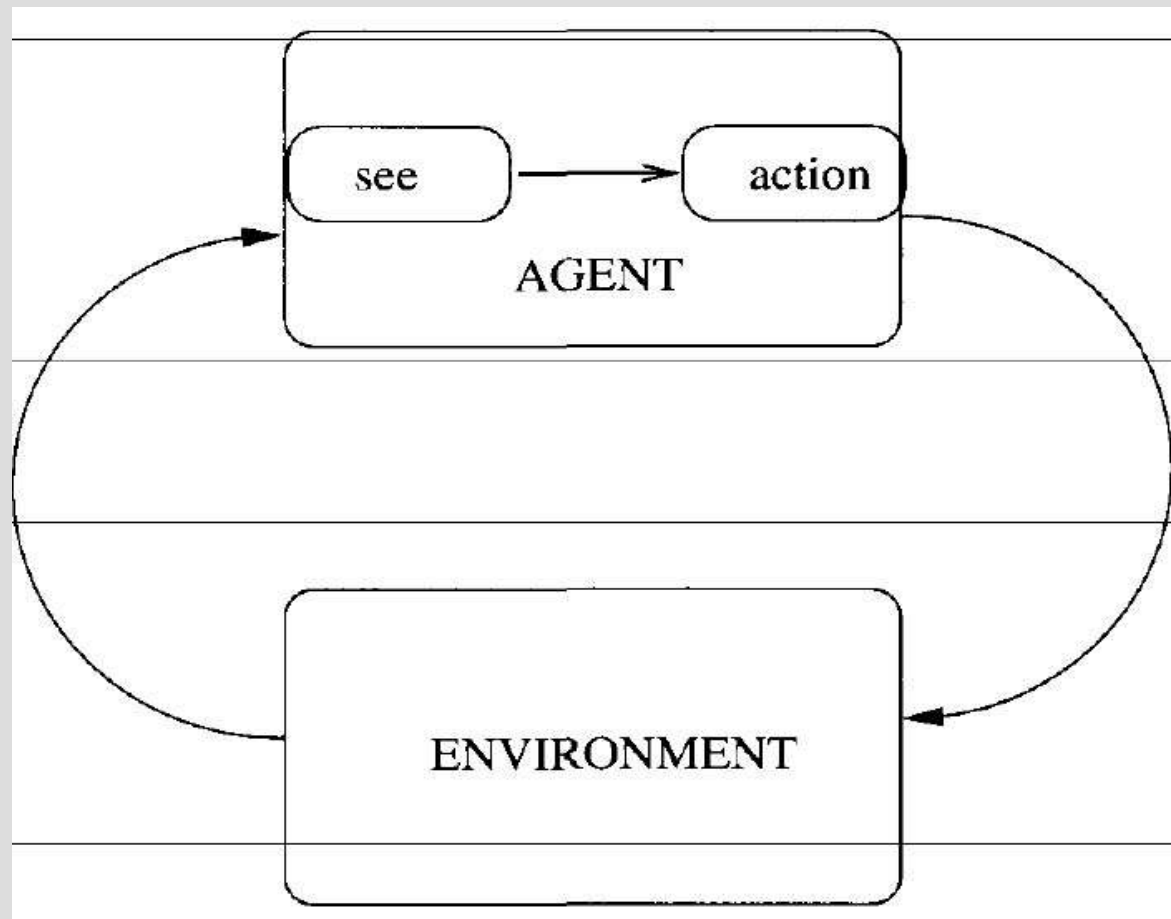
2.2 How can we make an Agent perceive?

- Basically, there are two approaches to the question of how to make an Agent perceive.
- In order to refine the model of an Agent, and so, making these Agents a little less daffy, we give the Agents some kind of subsystems.

2. Architectures of Intelligent Agents

2.2 How can we
make an Agent perceive?

- So, the structure looks like as follows:



2. Architectures of Intelligent Agents

2.2 How can we make an Agent perceive?

- The function **see** enables the Agent to receive input from the environment.
- In software Agents this could be implemented by commands of the platform that reveal information about the software environment.
- In hardware Agents, this is covered by perceiving hardware like cameras, microphones or sensors.
- Nevertheless, with **Per** being a (non-empty) set of percepts,

see: $E \rightarrow Per$

is the function which maps environment states to percepts.

2. Architectures of Intelligent Agents

2.2 How can we make an Agent perceive?

- So, we need a new function, that maps the percept to an output action.

action: Per \rightarrow Ac

- Now, an Agent can be considered a tuple

Ag = <see,action>.

2. Architectures of Intelligent Agents

2.2 How can we make an Agent perceive?

- Another approach is to give the Agents (in the very widest sense) some background knowledge
- That means, that Agents can remember previous states by referring to any kind of internal data structure.
- Then, an Agent has a set of internal states I , which influences the Agent's decision making process.
- The **see**-function is left unchanged, however the **action**-function has to be altered:

$$\mathbf{action}: I \rightarrow \mathbf{Ac}$$

2. Architectures of Intelligent Agents

2.2 How can we make an Agent perceive?

- And finally, we need function, that maps an internal state and a percept to another internal state. Hence:

$$\text{next: } I \times Per \rightarrow I$$

- We now can generate some algorithm like

<snip>

REPEAT

```
    initialize i0;                //Agent's internal state i0 initially
    see(e);                       //observe environment and generate percept
    i1 = next(i0, see(e));        //create next internal state
    action(next(i1,see(e)));      //choose and perform action
```

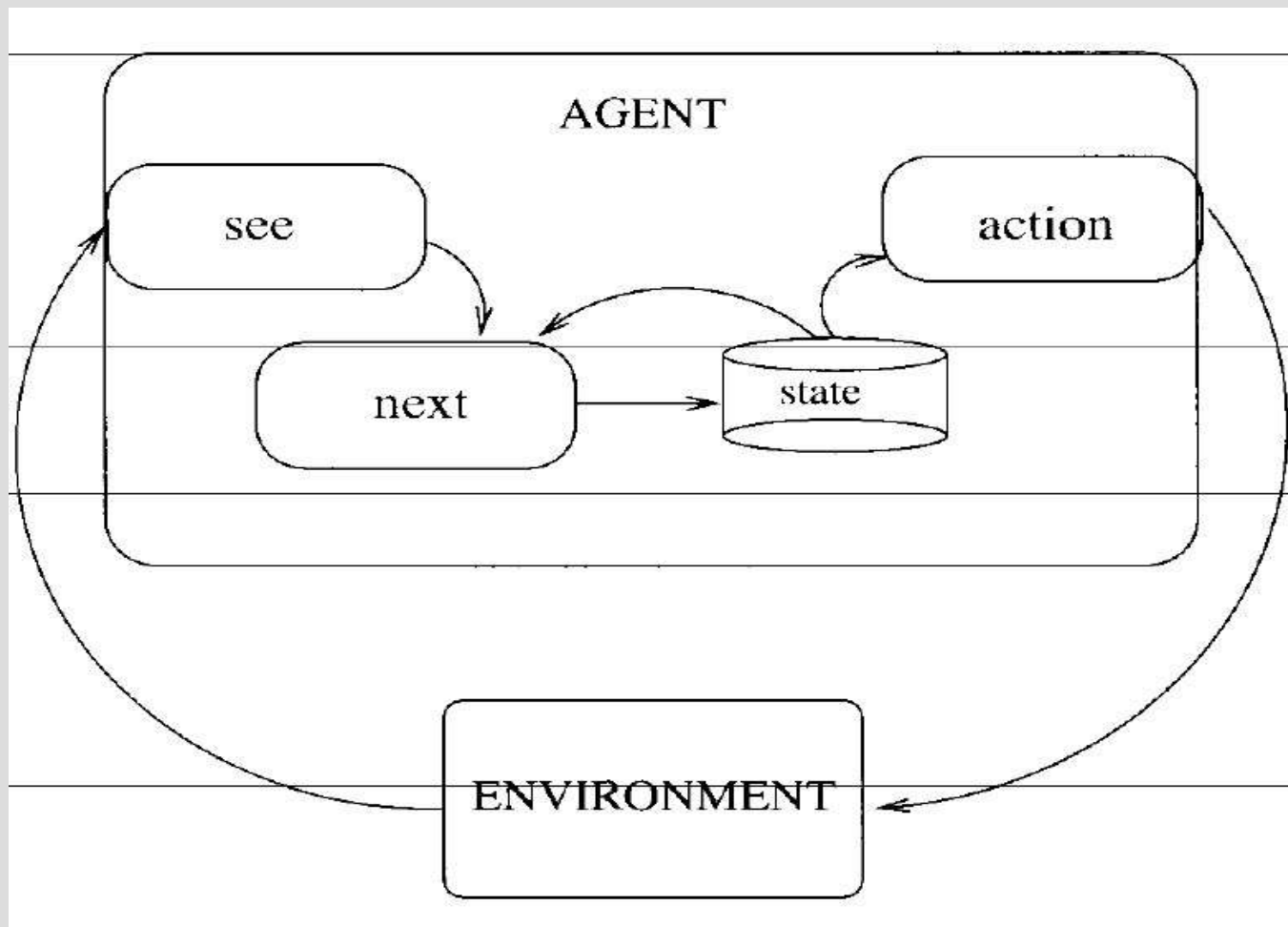
UNITL infinity ∞ or final state reached

</snap>

2. Architectures of Intelligent Agents

2.2 How can we
make an Agent perceive?

- And here is the diagram to such an architecture:



Overview

- 1. Overview
- 2. Architectures of Intelligent Agents – A Sketch
 - 2.1 How can Agent \wedge Environment interaction be formalized?
 - 2.2 How can we make an Agent perceive?
- **3. The Naming Game by Luc Steel – an example of a MAS**
 - **3.1 What is the Naming Game?**
 - **3.2 The Naming Game Formalisms**
 - **3.3 How does the Naming Game work?**
- 4. The Naming Game tweaked – a Research on MAS
 - 4.1 Intro and Method
 - 4.2 Results and Discussion

2. Architectures of Intelligent Agents

3.1 What is the Naming Game?

- How the world looks like
 - a population of agents
 - a set of unnamed objects
 - a set of possible names – created “on the fly”
- What the agents do
 - create a one-word-sentence language by
 - giving names to objects
 - negotiating these names
- The goal is
 - for all agents to speak the same language

2. Architectures of Intelligent Agents

3.2 Naming Game Formalisms

- Each Agent has an "Inventory", in which the relation between Object and Name is stored.

- $I_{a,t} \subseteq Q \times N \times [0.0, 1.0]$

- $Q \subseteq O$, the set of objects nameable by agent a
- N , the set of names
- $\gamma \in [0.0, 1.0]$, the strength of the relation

- WHAT?

- examples:

$$I_{1,0} = \{ \}$$

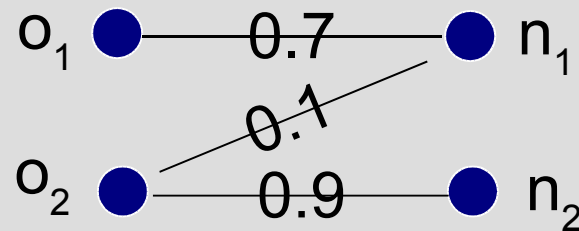
$$I_{1,4} = \{ \langle o_1, n_1, 0.7 \rangle, \langle o_2, n_1, 0.1 \rangle, \langle o_2, n_2, 0.9 \rangle \}$$

$$I_{2,4} = \{ \langle o_1, n_1, 0.4 \rangle, \langle o_1, n_2, 0.3 \rangle, \langle o_2, n_2, 0.5 \rangle \}$$

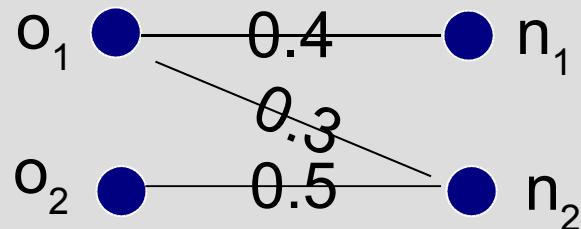
2. Architectures of Intelligent Agents

3.2 Naming Game Formalisms

- $I_{1,4} = \{ \langle o_1, n_1, 0.7 \rangle, \langle o_2, n_1, 0.1 \rangle, \langle o_2, n_2, 0.9 \rangle \}$



- $I_{2,4} = \{ \langle o_1, n_1, 0.4 \rangle, \langle o_1, n_2, 0.3 \rangle, \langle o_2, n_2, 0.5 \rangle \}$



2. Architectures of Intelligent Agents

3.3 How does the Naming Game work?

- Now, the Agents "negotiate" in each round.
- This is done, by
 - randomly choosing two Agents,
 - letting the speaker randomly choose an object,
 - letting the speaker say the name to the hearer,
 - letting the hearer choose, if the object is already named
 - updating the strength of the object-name-relation according to outcome of negotiation
- Recognize something? Can roughly be considered a GP approach, right?

2. Architectures of Intelligent Agents

3.3 How does the Naming Game work?

- What happens after one round?
- Depending on the outcome, there are 4 different cases
 - the speaker had no name for the topic
⇒ Invention
 - the hearer had no name for the topic
⇒ Adoption
 - speaker and hearer agreed on the name
⇒ Enforcement & Inhibition
 - speaker and hearer disagreed on the name
⇒ Damping

2. Architectures of Intelligent Agents

3.3 How does the Naming Game work?

- When speaker and hearer agreed on the name, everyone is happy.
 - Enforcement:
 - speaker and hearer both *increase* the strength of the association used for this success with Δ_{inc}
 - lateral Inhibition:
 - speaker and hearer both *decrease* the strengths of competitors
 - relations with another name for the same object are decreased with Δ_{n-inh}
 - relations with the same name for another object are decreased with Δ_{o-inh}

2. Architectures of Intelligent Agents

3.3 How does the Naming Game work?

- When speaker and hearer disagreed on the name, everyone is sad.
 - both speaker and hearer decrease the strength of the used association with Δ_{dec}
 - the speaker points to the object he/she/it meant
 - the hearer adopts this word

2. Architectures of Intelligent Agents

3.3 How does the Naming Game work?

- Please note, that Δ_{dec} , $\Delta_{\text{h-inh}}$, $\Delta_{\text{o-inh}}$, Δ_{inc}
as well as lateral inhibition are variables, that can be freely chosen by the user.
- Other variables are, for instance, can be Name Confidence ("bullheaded" hearer) and Agent Confidence ("bullheaded" speaker).

Overview

- 1. Overview
- 2. Architectures of Intelligent Agents – A Sketch
 - 2.1 How can Agent \wedge Environment interaction be formalized?
 - 2.2 How can we make an Agent perceive?
- 3. The Naming Game by Luc Steel – an example of a MAS
 - 3.1 What is the Naming Game?
 - 3.2 The Naming Game Formalisms
 - 3.3 How does the Naming Game work?
- **4. The Naming Game tweaked – a Research on MAS**
 - **4.1 Intro and Method**
 - **4.2 Results and Discussion**

2. Architectures of Intelligent Agents

4.0 The Naming Game Tweaked

- This is a Research Paper from the last semester.
- In this paper, the impact if several variables, the user can freely choose to alter, is analyzed.
- In order to do this, the Naming Game has been implemented...
- http://www.oswego.edu/~tenberge/cog366/lisp/mas/Paper_NamingGame.zip

2. Architectures of Intelligent Agents

4.1 Introduction and Method

- In theory, agents are meant to propose each other a name for a object and negotiate about that name.
- They shall find comprimises in naming these things and hence adapt to each other.
- This will on the long run establish a common language with one word for one object.

2. Architectures of Intelligent Agents

4.1 Introduction and Method

- Synonymy (many words for same object) was allowed...
- ... Homonymy (many objects with same name) wasn't.
- The impact of lateral inhibition, agent confidence and name confidence was investigated.
- All 8 combinations of these three switches were tested.

2. Architectures *of Intelligent Agents*

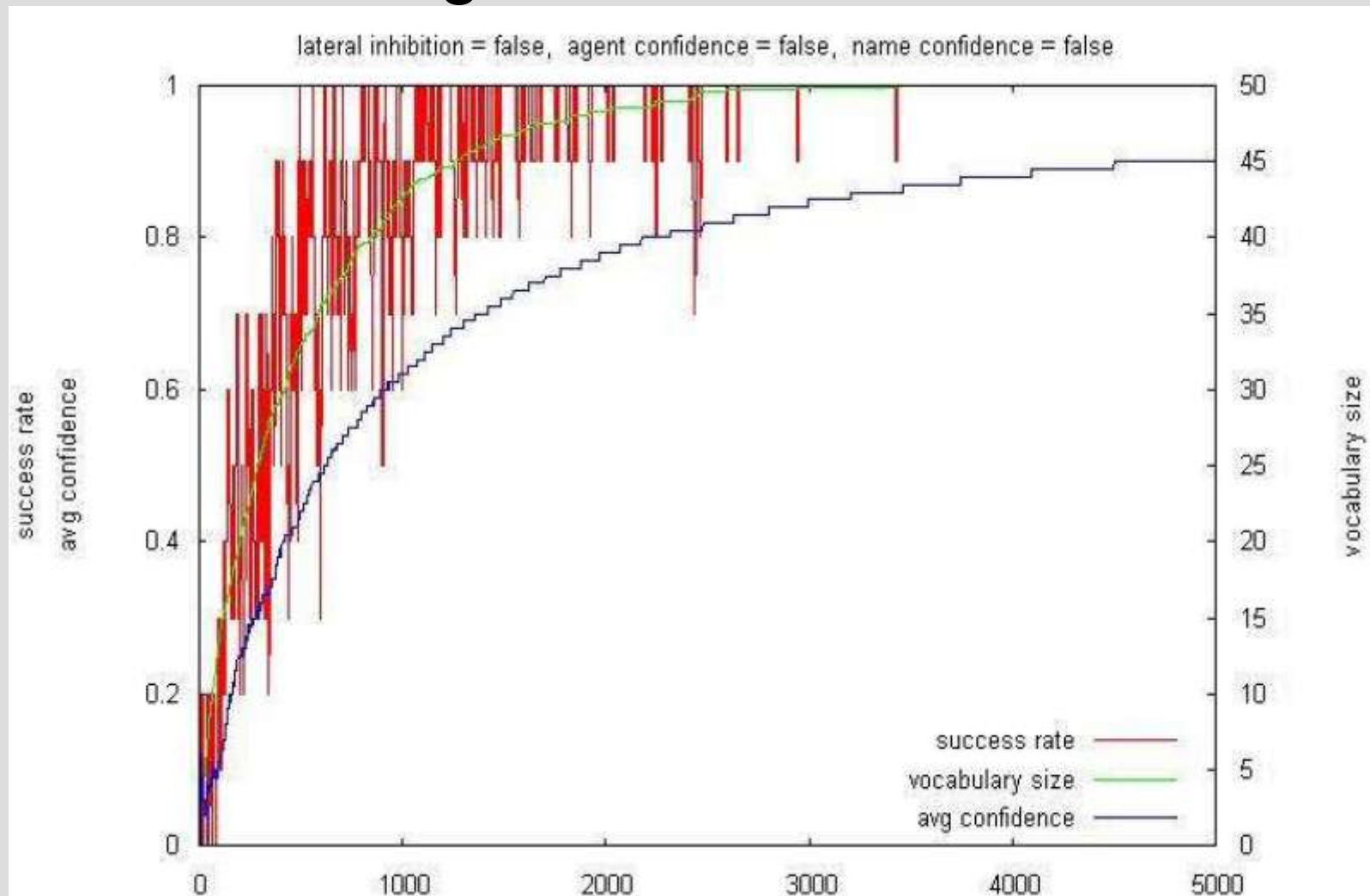
4.2 Result and Discussion

- The results showed, that even those artificial agents behave pretty much alike humans, that are involved in discussions.

2. Architectures of Intelligent Agents

4.2 Result and Discussion

- Here is the diagram, all switches turned off:



2. Architectures *of Intelligent Agents*

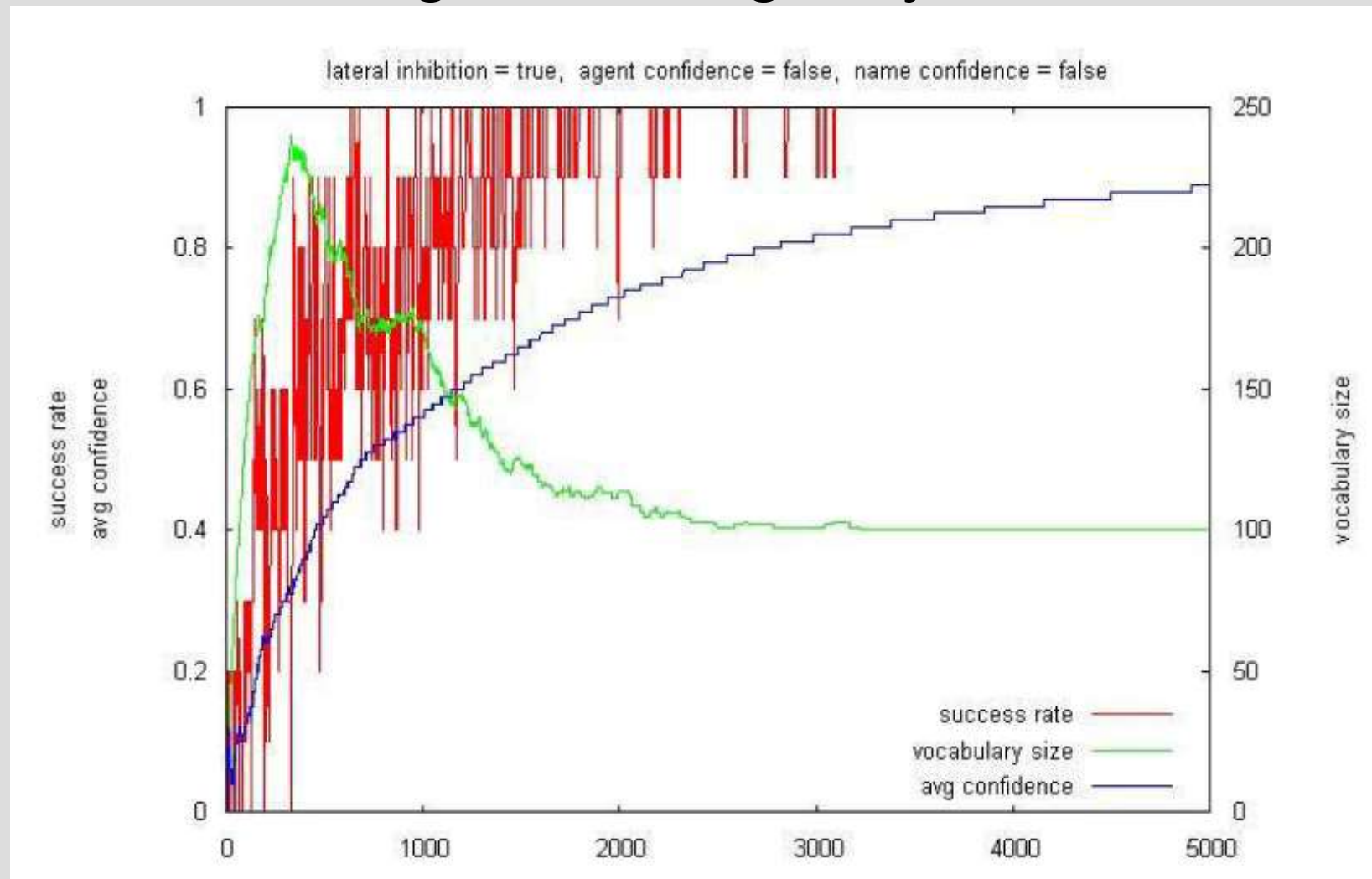
4.2 Result and Discussion

- The successrate becomes stable very late.
- An optimal vocabulary size cannot be reached.
- The average confidence is very low.

2. Architectures of Intelligent Agents

4.2 Result and Discussion

- Here is the diagram, using only lateral inhibition:



2. Architectures of Intelligent Agents

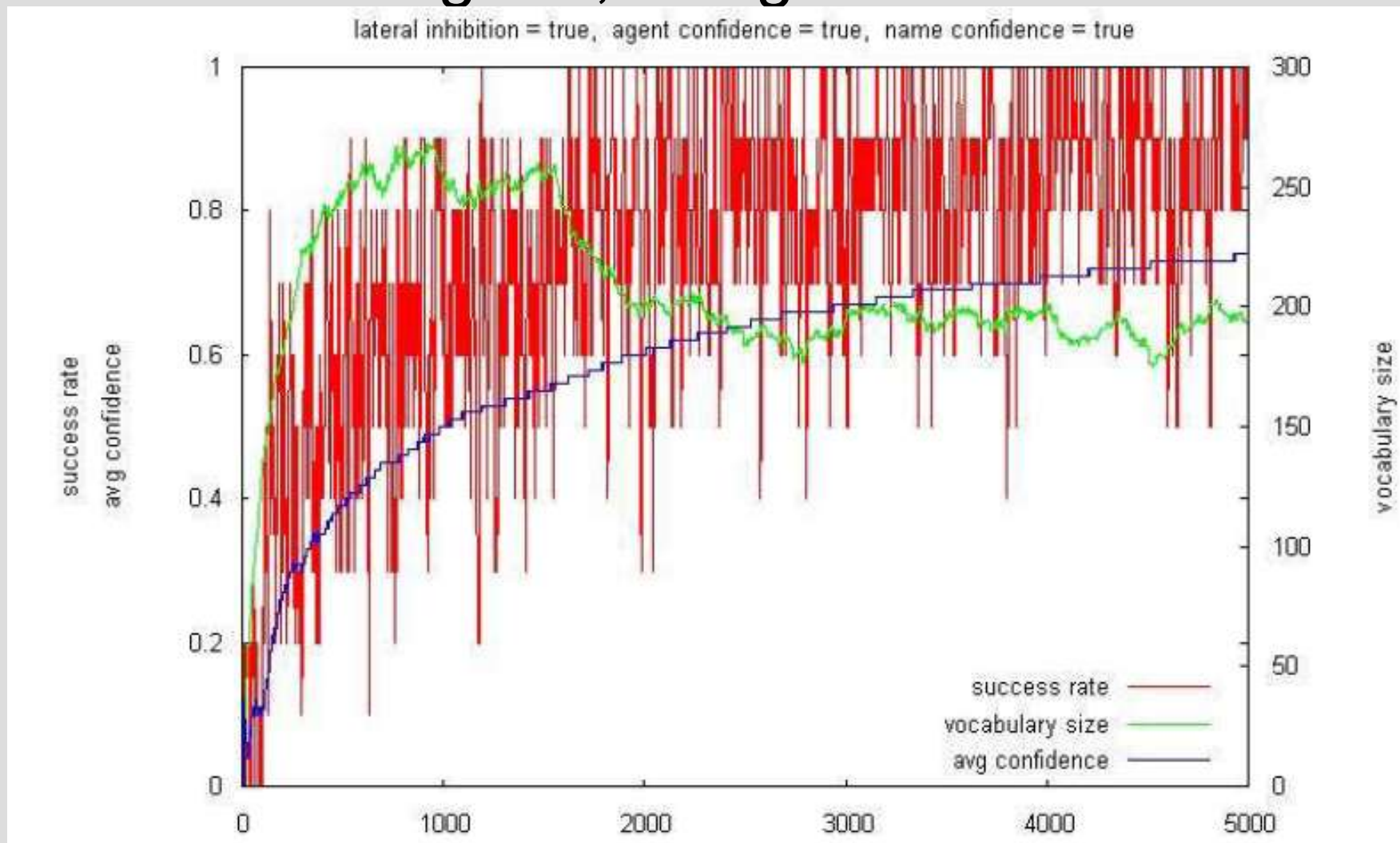
4.2 Result and Discussion

- The successrate becomes stable much sooner.
- An optimal vocabulary size is reached very soon.
- The average confidence is very high.
- "Wrong" knowledge is inhibited amongst the population, so that only that, what is globally regarded as correct, maintains.

2. Architectures of Intelligent Agents

4.2 Result and Discussion

- Here is the diagram, using all switches:



2. Architectures of Intelligent Agents

4.2 Result and Discussion

- The successrate is by far the most unstable one.
- An optimal vocabulary size is not reached at all.
- The average confidence is (surprisingly) high.
- Bullheaded agent behaviour makes a successful negotiation almost impossible.

2. Architectures *of* Intelligent Agents

The end

Any Questions?