"To avoid giving offence, I must here observe, that when I deny justice to be a natural virtue, I make use of the word natural, only as opposed to artificial. In another sense of the word, as no principle of the human mind is more natural than a sense of virtue, so no virtue is more natural than justice. Mankind is an inventive species; and where an invention is obvious and absolutely necessary, it may as properly be said to be natural as any thing that proceeds immediately from original principles, without the intervention of thought or reflection. Though the rules of justice be artificial, they are not arbitrary. Nor is the expression improper to call them Laws of Nature; if by natural we understand what is common to any species, or even if we confine it to mean what is inseparable from the species" (252). [1]

"But when men have observed, that though the rules of justice be sufficient to maintain any society, yet 'tis impossible for them, of themselves, to observe those rules in large and polished societies; they establish government as a new invention to attain their ends, and preserve the old, or procure new advantages, by a more strict execution of justice" (323).

David Hume, *A Treatise Of Human Nature,* Volume II, Edinburgh Edition, 1826

Modelling Hume - A Draft for HUME$_{1.0}$

January 13, 2008

1 Introduction

In *Of Morals* – that is part III of his Treatise of *Human Nature*(1739f.) – David Hume gives a draft of the origin of virtue and government. According to Hume, *both* are human *inventions*. They evolved and emerged in a long process that finally made it possible for us – i.e. mammal beings with a 'natural' nature that is more appropriate for living in *small* groups – to live together in *large* societies. Without any personal ties to almost all members of these large societies huge indirect exchange networks emerged. Embedded in them, based on division of labour and technical advancements of all sorts we enjoy, compared with our prehistoric predecessors that lived in small

[1] Note that Hume's *justice* essentially is respecting property, transfer of property by consent only, and keeping promises. Hume's justice is *not* (re-)distributional justice or a kind of fairness.

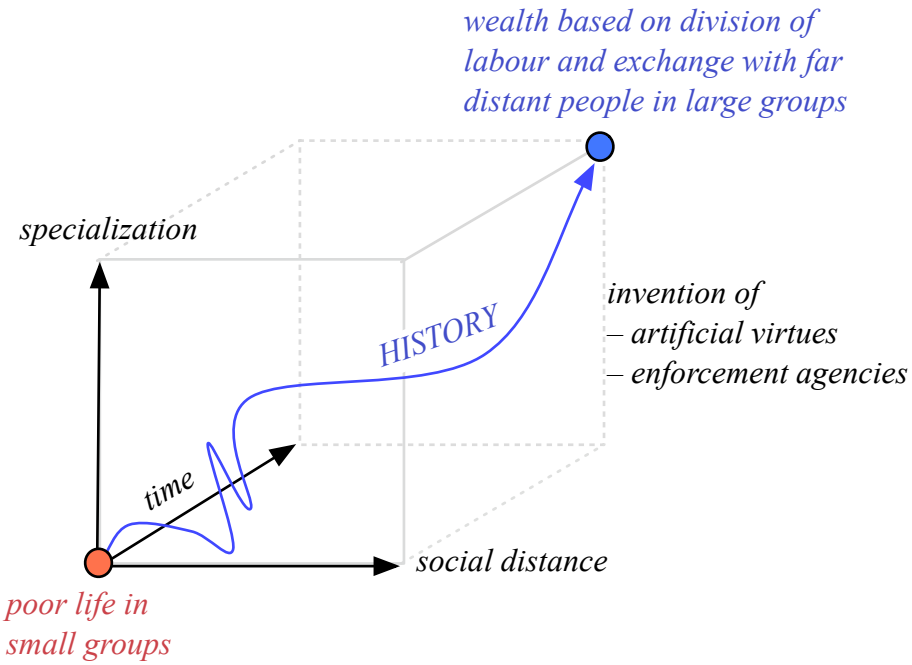

Figure 1: Hume's view of our history

groups, an unbelievable wealth – of course, not everywhere, not all the time, and not without setbacks, but in and for significant parts of the modern world and the modern times, at least the last two centuries[2] (see figure 1).

The question, how living together in large-scale societies is possible at all, puzzled already the ancient Greeks. In one of Plato's dialogs the sophist *Protagoras* gives a very modern answer which – after some deciphering of the myth in which it is couched (a myth about Prometheus and Epimetheus doing some creation work) – amounts to saying: A high blood toll was paid to learn the lessons. Then, finally, mankind invented both, *moral virtues* and *enforcement agencies*. That made possible to live together cooperatively in comparatively wealthy large-scale societies where high proportions of interac-

[2]Gregory Clark (*Farewell to Alms: A Brief Economic History of the World*, Princeton 2007) argues that up to 1800, there was basically a Mathusian economy in the world: For the average or median human being wealth tends to be on the subsistence level. This diagnosis is not totally new. New is the – now fiercely debated – explanation that Clark gives: Due to lucky circumstances that prevailed in England since about 1250, capitalist attitudes spread socially and probably even genetically from the top of society to the middle classes. That, finally, allowed to escape the Malthusian trap—so Clark's argument goes.

tions are no longer based on family ties or good personal acquaintanceship.[3] After deciphering, Protagoras' answer to the problem is – if not the same – at least very similar to Hume's draft. Key components in Hume's draft are:

1. an *original human nature* that – if not transformed and modified – in a very literal sense, causes serious trouble in large groups (for instance, confined generosity, favouring the loved ones and a systematic short-sightedness);

2. the invention of *'artificial virtues'*, especially justice, that are acquired by a 'moralizing' character transformation based on a kind of mutual moral training;

3. *division of labour* with a corresponding development of special capabilities;

4. the invention of *central authorities* that monitor, enforce, and – eventually – punish behaviour.[4]

Hume delivered a qualitative draft – more detailed and thought through than anything else at his time, nevertheless, a draft.[5] With HUME$_{1.0}$ we start to develop *a computational model of that draft*. Different from what Hume has done, the computational model will have precisely defined assumptions. Parameters, that are involved, will be explicit. That will allow, 'to play around' with the model, i.e. to seriously study the *interplay of a bunch of mechanisms*: We should be able to analyse *systematically* under what assumptions – in which parameter regions, more factual or more contrafactual ones – virtues, specialization, and wealth prosper—and how robust or how sensitive these processes are when parameters and/or mechanisms vary more or less. The model should allow, for instance, to analyse the 'critical' questions like, whether or not, living together in large societies presupposes an *underlying* small group structure – at least in the sense of a significant proportion of interactions with the same few and well known people.

The following (see figure 2) is a draft for the model HUME$_{1.0}$. That is a model, in which Hume's components (1)-(3) and especially virtues play an important role. However, central authorities – key component (4) – are *left*

[3]If we trust Lucretius' *De rerum natura*, then Epicurus must have held a similar view as well.

[4]One should note that establishing an enforcement agency is a special type of division of labour.

[5]For detailed analysis,reconstruction, and elaboration of Hume's draft cf. Hartmut Kliemt, *Moralische Institutionen – Empiristische Theorien ihrer Evolution*, Freiburg 1985; *Antagonistische Kooperation*, Freiburg 1986.

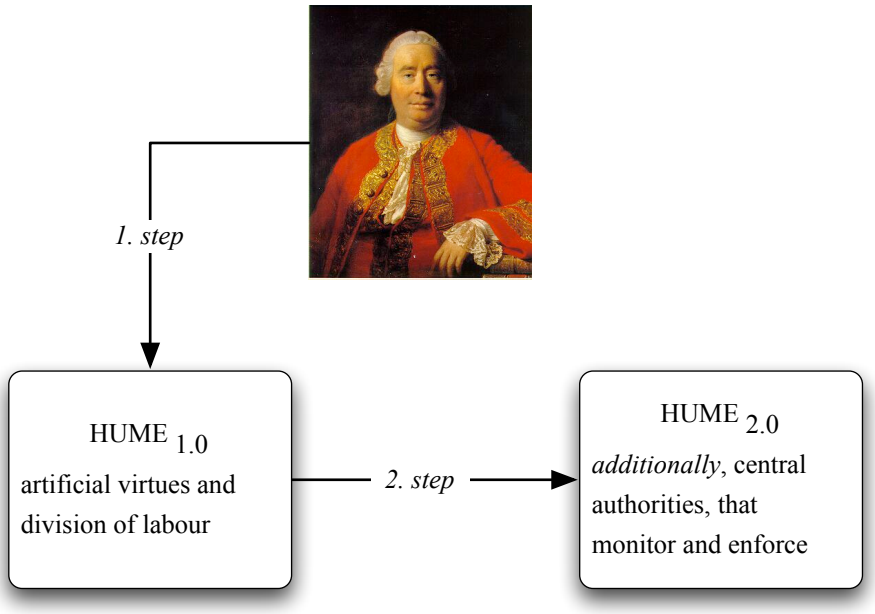

Figure 2: Modelling strategy: Two step heuristics

out. They will be included only in HUME$_{2.0}$. Thus, the model HUME$_{1.0}$ aims at an analysis of 'how far one can get' *without* central authorities.

2 Extended trust games as the basic games

Real life is a mix of games. Important ingredients of the mix are for instance: n-person prisoner's dilemma ($n \geq 2$), n-person-volunteer's dilemma, assurance games, coordination games or ultimatum games.[6] Modelling Hume we focus at the beginning on one and only one game: The *trust game* (TG). The game is very simple 2-person game. It plays a central role in Hume's analysis of trust and promises. TG is given – in extensive form – by the game tree in figure 3. Since the utilities are only unique up to positive affine transformations, two payoffs can be set to 0 and 1 respectively. As a consequence, then, all possible TGs can be characterized by the two parameters a and b. To keep the characteristic rank ordering of the results of TGs, the restrictions are that a<0 and b>1.

[6]Probably from Stone Age times to now, humans (and may be other animals as well) have to cope with games as listed here. However, the distribution of the types of games may have changed over times and societies.

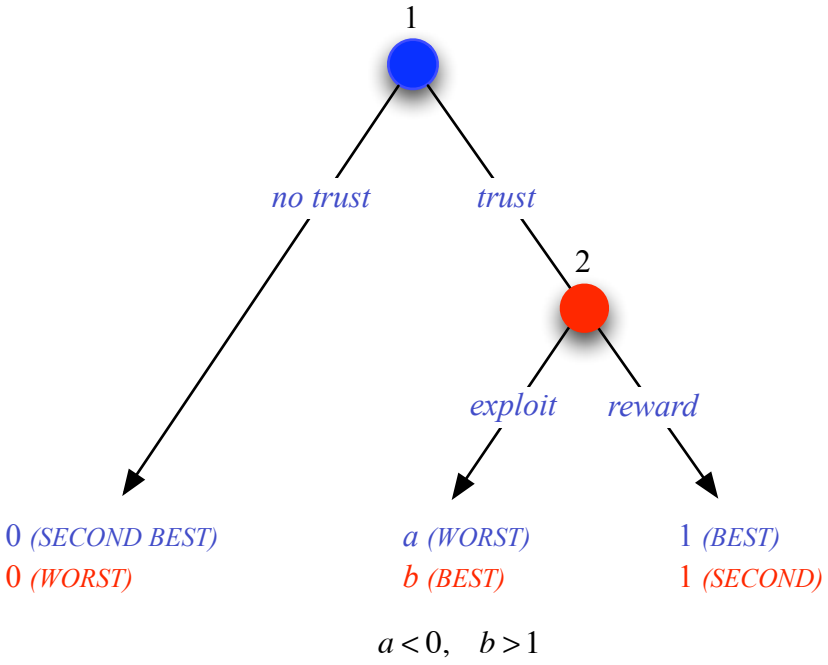

Figure 3: The simple trust game (TG)

It is trivial to see by backward induction that for *rational* players the *noniterated, one-shot* TG has only one solution – and that is an inefficient one: Anticipating that player$_2$ will go for exploitation at his node, player$_1$ decides for *no trust* at his very first decision node. Result is an inefficient outcome with the payoff $\langle 0, 0\rangle$. If we assume an iterated TG, the result *might* be mutual trust in all basic games, given the probability α for a continuation of the game is high enough with respect to the involved payoffs. For instance, a sub-game perfect Nash-equilibrium could consist of a pair of super-game strategies in which both players start 'co-operatively' and punish the other one by eternal retaliation after a very first 'defection' ($TRIGGER$-strategy).[7]

The interaction structure captured by the trust game is the very core of our model—though in an *enriched setting*. To give a feeling, key ingredients of the setting are outlined in a *first* step. Then, in a *second* step, details are worked out.

- The agents live in a world in which one may get *special problems*.
- By solving problems agents develop *special competencies* that affect

[7] TRIGGER-strategies are important: If they 'do not work', then *nothing* works. An equilibrium based on TRIGGER-strategies is therefore a kind of base line equilibrium. – Under the conditions in figure 3 and a continuation probability α, the payoff Π^{reward} that player$_2$ can *expect* by rewarding all the time, is

$$\Pi^{reward} = 1 + \alpha \cdot 1 + \alpha^2 \cdot 1 + \ldots + \alpha^{t-1} \cdot 1 + \ldots = \sum_{t=0}^{\infty} \alpha^t = \frac{1}{1-\alpha}$$

If, alternatively, player$_2$ *exploits* in the first game and – consequently – player$_1$ retaliates forever, then the expected super-game payoff for player$_2$ is:

$$\Pi^{\exp loit} = b + \alpha \cdot 0 + \alpha^1 \cdot 0 + \ldots \alpha^{t-1} \cdot 0 + \ldots = b$$

After some lines of algebra, one gets that

$$\Pi^{reward} \geq \Pi^{\exp loit} \Leftrightarrow \alpha \geq \frac{b-1}{b} = \alpha^*$$

Thus, if the continuation probability α is not smaller than the exploitation-payoff dependent *threshold* value $\alpha^\star$, then there is no incentive for player$_2$ to switch to a super-game strategy that exploits in the *first* move. That threshold does not change if we consider a super-game strategy that exploits *later*. Since player$_1$ can't get more than by a rewarding player$_2$, he has no incentive to switch to another supergame strategy either. Thus, we have an *equilibrium*. Note: This equilibrium depends only on a certain payoff of player$_2$ (and his time-preference if we interpret α not as a continuation probability rather than a time preference – though I do not favour that possible interpretation). – For a detailed and careful analysis of the trust game and its importance to understand Hume's *Treatise* and *Enquiry* cf. Bernd Lahno, *Versprechen – Überlegungen zu einer künstlichen Tugend*, München 1995.

both, the *value* of their solutions and the *costs* to produce them.

- An agent with a certain problem might therefore look for a more competent other to solve it in a cheaper and better way. Thus, pairs of players are not simply given, rather than result of a *matching* process. (In that process, the competence and trustworthiness of agents plays an important role. The details will be explained later.)

- If a match is established the agent with the problem (player$_1$) has to do some *prepayment* in whatever currency: crop, prey, personal service, i.e. valuables of all sorts. Only afterwards the agent that was 'hired' to solve the problem (player$_2$), starts working on the solution—*or not*.

- Prepayment of player$_1$ and the resulting *temptation* for player$_2$ to keep the prepayment without delivering the solution, makes our enriched setting a kind of *extended version* of the original trust game (ETG).

Key concepts and features of the enriched setting –for instance, problem assignment, competencies and their dynamics, competence dependent costs and values– need further elaboration before they can be implemented in HUME$_{1.0}$:

1. Period by period at least some of the agents get a *problem*. Problems are characterised by positive integers $k \leq K$, with K being the exogenously given upper limit for the number of problems (and sometimes the set of them – constant over time). Among their characteristics, agents have a time dependent competence vector with K components. The real valued kth component characterises the competence c_{ik} of an agent i to work on problem k. At the beginning, all agents are equally bad/good[8] in solving certain problems, i.e. $c_{ik} = \frac{1}{K}, \forall i \forall k$. By working on a *certain* problem, agents become better at solving the type of problem they are working on. However, at the same time their other competences *deteriorate*. Formally, that can be realized by, *firstly*, adding a certain Δ to the component in question and, secondly, renormalizing the whole competence vector in such a way that again $\sum_{k=1}^{K} c_{i,k} = 1$ holds. – In the ETG, the player$_1$ of the simple TG is an agent with a certain problem; we will therefore often refer to player$_1$ as the P-player or the P-agent.[9] The P-player's problem $k \in K$ is randomly assigned. Player$_2$

[8]It could be interesting to analyze whether or not and to what degree small variations in competencies already for $t = 0$ matter afterwards.

[9]This coinage is also to allude to the *principal-agent problem*. If competencies are not observable, or, given the competence, effort levels matter, then the situation has the decisive information asymmetries that constitute a principal-agent problem.

is the one to work on the problem and thereby to solve it for player$_1$; we often refer to player$_2$ as the S-player or the S-agent. "player$_1$" and "player$_2$" refer to the roles of agents in the game. If a more absolute reference is necessary, we will refer to the players as agent i, agent j or agent l.

2. The actual problem k assigned to the P-player i and the S-player's j competence c_{jk} in solving it, affect the payoffs in the extended trust game. In general, competence affects both, the costs and the value of a solution which, then, determine the payoffs in the *ETG*.

   (a) The *higher* the competence, the *higher* the value of the solution. Infinitely many functions could could be used to describe the intended effect. We use the very simple non-linear function

   $$\text{VALUE}_{jk} = constant_{value} + c_{jk}^{\varphi}$$

   VALUE_{jk} is the value of the solution of problem k if agent j with the actual competence c_{jk} works on the solution. The exponent φ controls the strength of the '*competence effect*': For $\varphi < 1$ the value of the solution increases steep with increasing *low* competencies; for $\varphi > 1$ the value increases steep with increasing *high* competencies. For $\varphi = 1$ the value is proportional to competence. The $constant_{value}$ shifts the function upward and downward. Figure 4 shows the effects for different values of φ under the condition $constant_v = 1$.

   (b) The *higher* higher the competence, the *lower* the costs to produce the solution. To determine costs we use – following the same 'spirit' as above in (a) – the function

   $$\text{COSTS}_{jk} = constant_{costs} - c_{jk}^{\sigma}$$

   It is now the exponent σ that controls the strength of an over- or under-proportional competence effect, while $constant_{costs}$ moves the function upward and downward.

   Figure 5 shows the *costs* for different values of σ, now under the condition that $constant_{costs} = 1$.

   (c) Value minus costs gives the value *added*. To simplify the situation we will assume from now on that $constant_{costs} = constant_{value} = 1$. Based on that we get

   $$\text{VALUE\_ADDED}_{jk} = \left(1 + c_{jk}^{\varphi}\right) - \left(1 - c_{jk}^{\sigma}\right) = c_{jk}^{\varphi} + c_{jk}^{\sigma}$$

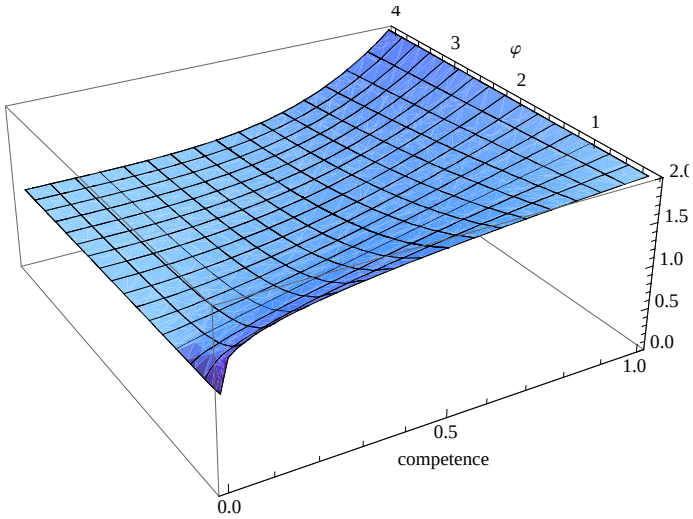

Figure 4: Competence dependent *value* of a solution($0.25 \leq \varphi \leq 4$) under the condition $constant_{value} = 1$.

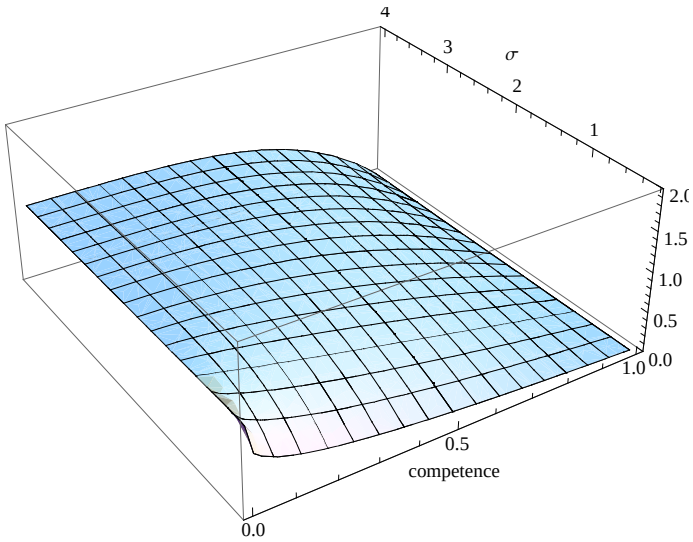

Figure 5: Competence dependent *costs* of a solution($0.25 \leq \sigma \leq 4$) under the condition $constant_{costs} = 1$.

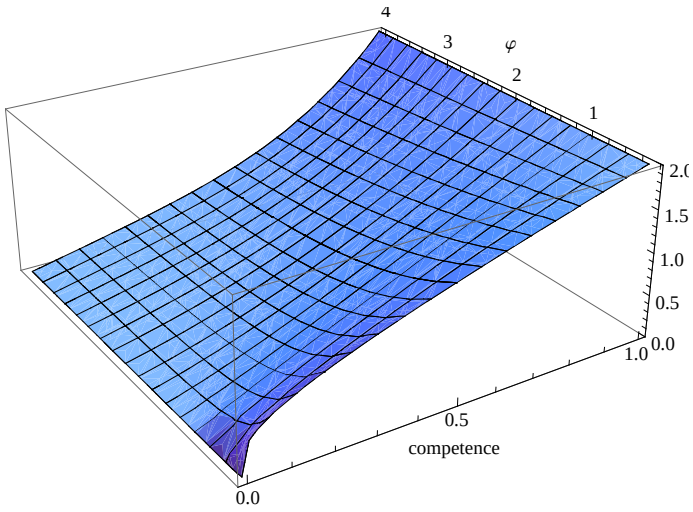

Figure 6: Competence dependent *value added* for $0.25 \leq \varphi \leq 4$ and $\sigma = 1$

The simplifying assumption has certain effects: Positive competencies generate positive values. We avoid negative added values. For a zero-competence, costs are 1 and the added value is 0. For a competence 1, costs are 0 (but note that the dynamics of competencies as described above does *not* lead to a competence that is 1—though agents can get closer and closer to that) and the added value[10] is 2.

Figure 6 shows for $\sigma = 1$ the *value added* for different competencies and values of φ.

3. The enriched setting and the elaboration given so far allows for *different* types of interactions that *all* involve serious trust problems. There are at least three –highly stylised– situations:

   (a) In a *first* type of extended trust game, period by period a certain number of randomly selected agents becomes P-agents, i.e. agents with randomly assigned problems. *All others* are S-agents. P-agents try to find competent and trustworthy S-agents that could solve their problem. Figure 7 gives the details of this first variant of an extended trust game. A trust problem is induced by step 5: P-agent i has to *prepay* j's costs plus a certain fraction β of the

[10]Obviously, the assumption $constant_{costs} = constant_{value}$ makes 2 the maximum added value that is possible. We have to be aware of that fact. It may well be the case that that restriction has to be changed in view of other components of the model—for instance with regard to learning mechanism that may react sensitive on the size of payoffs.

value added by S-agent j. Since in the following step 6 the S-agent j is, firstly, free to exploit or to reward and, secondly, better off by exploitation, we get a fully fledged trust problem.

In the structure just described, S-agents do not have problems. But they can work on solutions. P-agents, on the contrary, can't work on solutions—not even solutions of their own actual problem. The latter may be allowed in a slightly modified structure: In step 4, – a case in which i did not find a competent and trustworthy other – the P-agent may resort to his *own* competence to solve his problem. If that option should exist, then we have, firstly, to design the matching procedure applied in step 2 accordingly. Secondly, the payoffs have to be adapted: If P-agent i follows the 'Do it yourself!'-devise, he gets as payoff the added value based on his own competence, i.e. VALUE_ADDED$_{ik}$—perhaps minus some costs for the futile search.

To have a short reference to the type of situation captured by the extended trust game here described, we will refer to it as the *ETG/search for solutions/* or – shorter – $ETG_{solutions}$.

(b) In a *second* type of situation *everybody* gets a problem and *everybody* starts working on it—though within the limits of his competence. After having done that, everybody may look for more competent others to do some *refinement* work on the 'preliminary solution'. If a (seemingly) trustworthy and competent refiner is found in a matching process, the provisional – though not worthless – preliminary solution is handed over to the refiner. The refiner gets as a prepayment the preliminary solution plus a certain fraction β of the added value he might generate—or not. We will refer to this type of extended trust game as *ETG/search for refinements/* or – shorter – $ETG_{refinements}$.

For the value that is added to the value of agent i's preliminary solution of problem k by agent j's refinement work, we assume:

$$\text{VALUE\_ADDED\_BY\_REFINEMENT}_{jik} = (\text{VALUE}_{jk} - \text{VALUE}_{ik}) - \text{COSTS\_REFINEMENT}_{jik}$$

But what are exactly j's costs when doing the refinement work on i's solution of problem k? We will make two assumptions: *Firstly*, the costs are only a fraction of the costs j would have if j had to start from the scratch. *Secondly*, the refinement costs are proportional to

$$\frac{\text{VALUE}_{jk} - \text{VALUE}_{ik}}{\text{VALUE}_{jk}}$$

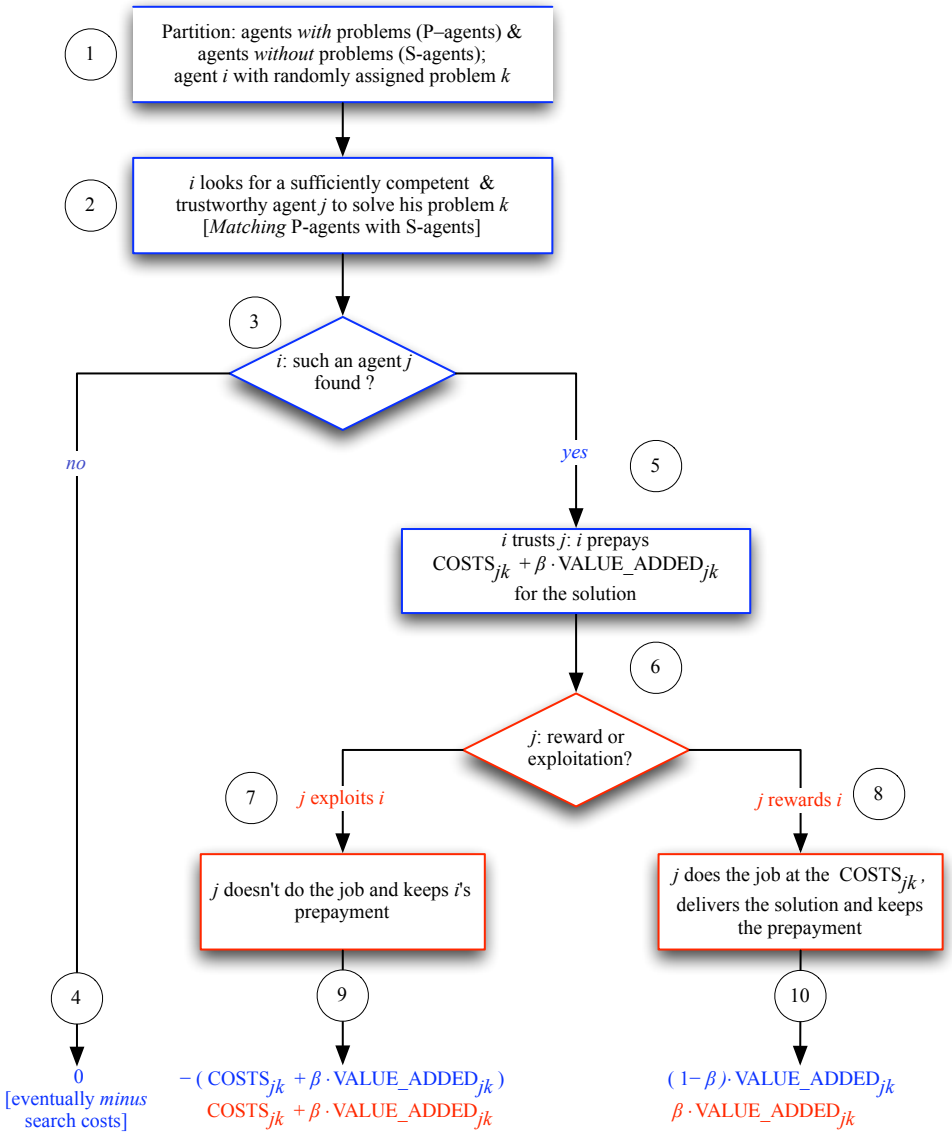

Figure 7: Extended trust game, first version: $ETG_{solutions}$. *Blue*: Action or payoff of agent *i*. *Red*: Action or payoff of agent *j*.

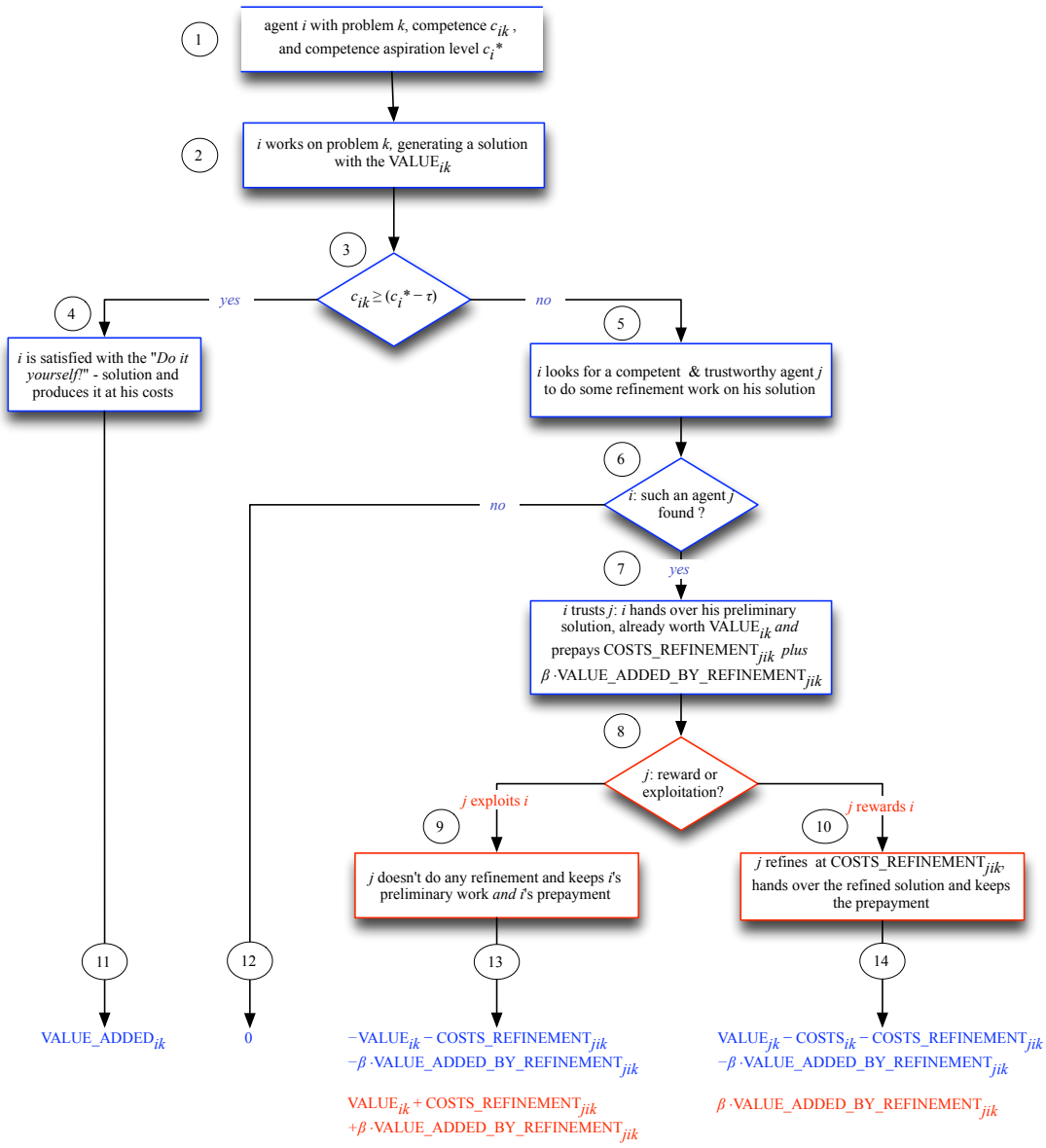

Figure 8: Extended trust game, second version: $ETG_{refinements}$. *Blue*: Action or payoff of agent i. *Red*: Action or payoff of agent j.

Thus, the total refinement costs for j if working on i's preliminary solution of problem k are

$$\text{COSTS\_REFINEMENT}_{jik} = \frac{\text{VALUE}_{jk} - \text{VALUE}_{ik}}{\text{VALUE}_{jk}} \cdot \text{COSTS}_{jk}$$

By substitution we get

$$\text{COSTS\_REFINEMENT}_{jik} = \frac{c_{jk}^{\varphi} - c_{jk}^{\sigma}}{1 + c_{jk}^{\varphi}} \cdot \left(1 - c_{jk}^{\sigma}\right)$$

Consequently, after some lines of algebra we then get for the value added by refinement

$$\text{VALUE\_ADDED\_BY\_REFINEMENT}_{jik} = \frac{\left(c_{jk}^{\varphi} - c_{ik}^{\varphi}\right)\left(c_{jk}^{\varphi} + c_{jk}^{\sigma}\right)}{1 + c_{jk}^{\varphi}}$$

This value is always *positive* if agent j is more competent than agent i, i.e. $c_{jk} > c_{ik}$.

More details for the second type of interaction are given in figure 8. In step 7 the β has a similar meaning as in $ETG_{solutions}$: agent i has to *prepay* j's costs plus a certain fraction β of the value added by j's refinement. In step 12 i did not find a competent and trustworthy refiner and receives – similar to the first version of the ETG the payoff 0. Alternatively, one might again allow that i resorts to his *own* competence to solve his problem. (As a consequence, we have to redesign the matching procedure and to redefine the payoff for this case: If P-agent i follows the 'Do it yourself!'-devise, he gets as payoff the added value based on his own competence, i.e. VALUE_ADDED$_{ik}$—perhaps minus some costs for the futile search.)

In this second type of situation (we refer to it as $ETG_{refinements}$) obviously in every period everybody can work on *two* jobs: preliminary work on his own problem *and* refinement work on the preliminary work of someone else.

(c) In a *third* type of situation – again – *everybody* gets a problem. But different from $ETG_{refinements}$ nobody has at first to start working on his own problem: Everybody can work on *one and only one* problem, *either* his own *or* on the problem of somebody else. The exchange of solution does *not* need to be directly reciprocal: It may be well the case, that – utilising their comparative advantages

– agent j works on the problem of agent i, while agent i works on the problem of agent l. The only restriction is that an agent works on just one problem per period.[11] We will refer to this variant of an extended trust game as $ETG_{exchanges}$. Details are given in figure 9.

As an alternative to step 11 – the case in which i did not find a competent and trustworthy refiner and receives the payoff 0 – we might again allow that i resorts to his *own* competence to solve his problem.[12]

Thus, the three stylised situations, we want to analyse *differ* under the perspectives:

- Do all agents have a problem? In $ETG_{refinements}$ and $ETG_{exchanges}$ all agents have a problem; in $ETG_{solutions}$ it is only a subset.
- Do the agents have to work on their own problem? In $ETG_{refinements}$ all agents have to do some preliminary work on their own problem; in the basic version of $ETG_{solutions}$ agents can work only on problems of others. In $ETG_{exchanges}$ agents can work on one problem only, either a problem of others or their own problem.
- Can the agents resort to a 'Do it yourself'-solution if they failed to establish a match? In the basic versions of $ETG_{solutions}$, $ETG_{refinements}$ and $ETG_{exchanges}$ they can't, but it would be easy to include such an option.

All three types of extended trust games have – besides the usual *trustworthiness* – at least *two additional 'moral' dimension: Firstly*, given that the competence vector is *not* directly observable, then there might be an incentive to be *deceptive* about one's competence – simply, to get the job via the matching process. *Secondly*, if we do not take for granted that a S-agent delivers with exactly the quality his competence allows for, then *underperformance* may be tempting for the S-player's: Normally quality is costly for the producer.[13] Thus, in the extended trust game a total of *three* moral qualifications, norms or virtues is involved. All of them regard the S-player:

[11] In later versions of that restriction may be given up.

[12] Again, if we do that, we have to redesign and adapt the matching procedure and set the payoff to VALUE_ADDED$_{ik}$—perhaps minus some costs for the futile search.

[13] Formally, we could introduce a performance level l_2 with $0 \leq l_2 \leq 1$ that represents to which degree player$_2$ is doing his best. The reward payoff for player$_1$ with problem k then is $1 + s \cdot c_{2k} \cdot l_2$.

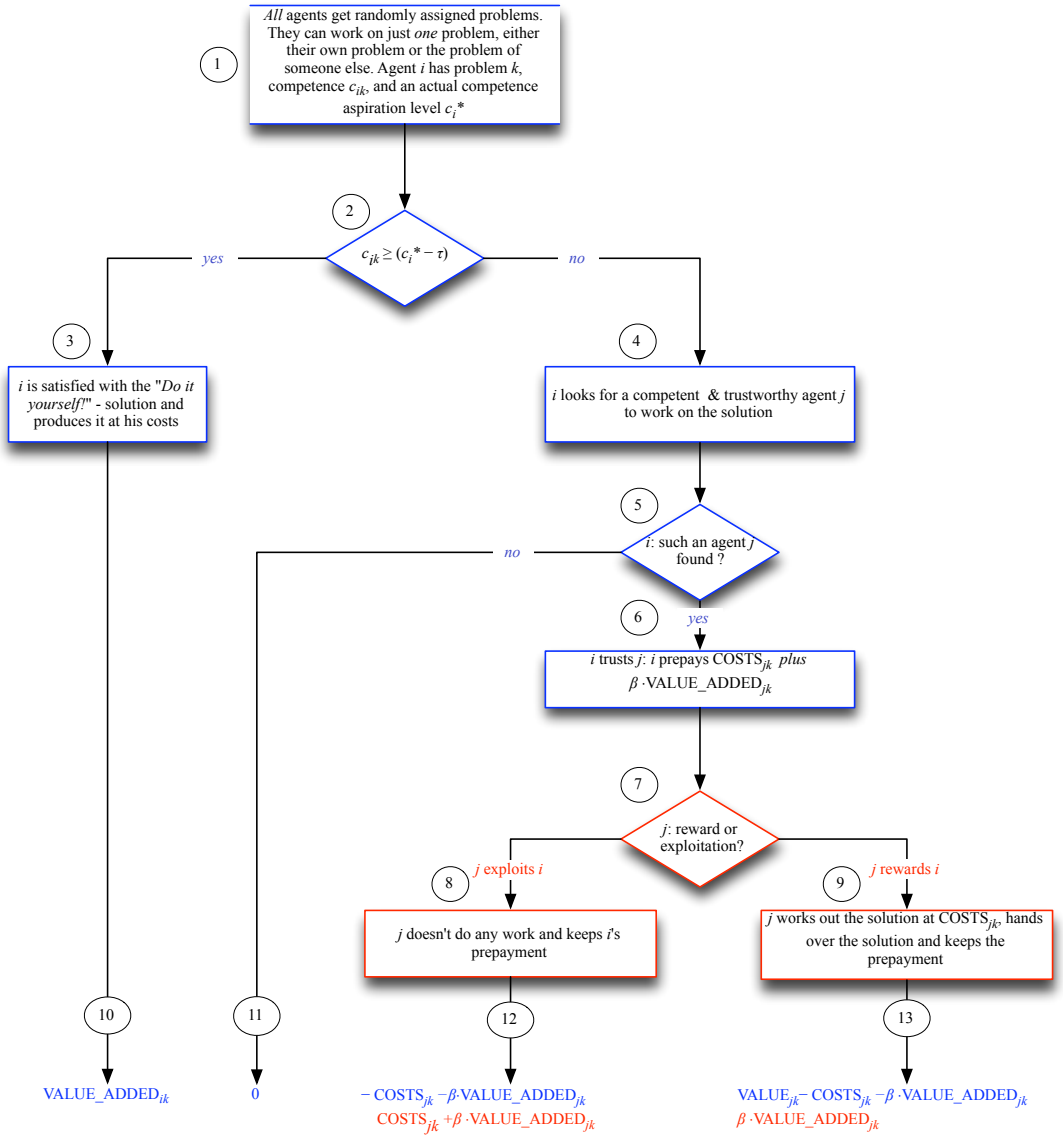

Figure 9: Extended trust game, third version: $ETG_{exchanges}$. *Blue*: Action or payoff of agent i. *Red*: Action or payoff of agent j.

1. *Trustworthiness*: Does player$_2$ reward?

2. *Honesty*: Does player$_2$ really have the announced technical competence in solving problem k?

3. *Reliability*: Does player$_2$ really do his best to deliver with exactly the quality his competence allows for?

Though the *ETG*s have all these moral dimensions, we will *simplify* the situation in HUME$_{1.0}$ by two assumptions: *Firstly*, we assume *error free direct observability of competencies*. Thus, deception – including *self*-deception – is excluded. *Secondly*, we forget about the reliability dimension and assume that *always the best quality is delivered*, that an agent's competence allows. By this assumption, we obviously *avoid* to deal with problems of effort detection right at the beginning. (Soon one will see that there are more than enough serious detection problems left we have to solve right at the beginning.) As a consequence HUME$_{1.0}$ focuses on the dynamics of trustworthiness, but in a social setting in which agents face special problems, special competences for solving special problems can be developed, competences matter in terms of payoffs and exchange partners have to be found.

A *methodological* remark: For the description of the *TG* we used an orthodox game tree. The *ETG*s are described by flow charts—and that even partially only. In a strict game theoretic description of an *ETG* the game trees would be a part of a dynamic matching game that is by far too complicated to be explicitly described by orthodox methods. The prospects of applying game theoretic solution conscepts to a well-specified reasonably complete non-co-operative game model seem even worse.

3 The basic setting: Two structural scenarios

HUME$_{1.0}$ analyses the *ETG*s in two different structural scenarios. The differences regard especially two questions: *Firstly*, whether or not the population of agents is partitioned into non-overlapping subgroups or not; *secondly*, whether or not *grid* based network structures affect possible interactions, the spread of information etc. In both scenarios, it is assumed that period-by-period nature randomly assigns a problem $j \in P$ to half of the population. Thus, one half of the population are P-agents, the other half are S-agents.

First setting is a grid distance based scenario (GD-scenario). The basic structure of the GD-scenario is: Agents live on a low-dimensional *grid* (1-dimensional, 2- dimensional, rectangular, hexagonal, irregular in the sense of

Voronoi-diagrams) with *overlapping neighbourhoods*. The core of the moral dynamics regards a *distant dependent trustworthiness*. The decisive details are:

1. Within a certain agent-relative, dynamical *radius*, defined in terms of network distance, P-agents look for *trustworthy* S-agents, which are, – given the P-agent's actual problem – at the same time are as *competent as possible* S-agents. We refer to the critical radius as the P-agent's *search radius* $radius_i^P(t)$. (The details of the matching process and the dynamics of the search radius – the latter driven by various kinds of learning – will be explained later.)

2. S-agents reward with a certain agent-relative and dynamical probability that *decreases with distance* – distance again understood as network distance. There are lots of linear or non-linear functions that could be used. In HUME$_{1.0}$ a simple *step*-function will be used. The critical distance is given by $radius_i^S(t)$: We refer to the reward probability *within* the (time dependent!) radius as $p_i^{reward \leq radius}(t)$, while $p_i^{reward > radius}(t)$ is the probability to reward beyond that radius. The probabilities for *exploitation* are then defined as

$$p_i^{exploit \leq radius}(t) = 1 - p_i^{reward \leq radius}(t)$$

and, correspondingly, as

$$p_i^{exploit > radius}(t) = 1 - p_i^{reward > radius}(t)$$

3. At the beginning we will assume that *within* the radius $radius_i^S(t)$ S-agents reward for sure, *beyond* that radius, they exploit for sure. Later, in HUME$_{1.X}$, we will allow for reward probabilities *different* from one or zero. Then, the radius *and* the two probabilities have a dynamics driven by some learning – to be explained later.

The framework allows – that is the hope – to study what one might call the *expanding-circle-effect*[14]: In a long historical process humans learn to be trustworthy towards far distant people.

The *second setting* is a **p**artition and **m**arket based scenario (PM-scenario). The basic structure of the PM-scenario is the following: The whole population is *partitioned into groups*. The agents have the choice whether to go for an extended trust game *either* within their group or on an open market, i.e.

[14]Cf. Peter Singer, *The expanding circle – Ethics and Sociobiology*, New York 1981.

a central point where all agents which decided to play outside their group meet, get matched and – if so – play an extended trust game. The decisive structural details are:

1. With a certain agent-relative, dynamical probability, P-agents enter the market and look there for trustworthy and at the same time – given the P-agent's actual problem – as competent as possible S-agents. The probability is $p_i^{market}(t)$. Correspondingly, with probability

$$1 - p_i^{market}(t) = p_i^{local}(t)$$

   agent i looks locally, within his group, for a trustworthy and competent S-agent. (The learning driven dynamics of the probabilities will be explained later.)

2. S-agents *reward* with a certain agent-relative and dynamical *probability*: With probability $p_i^{reward\_market}(t)$ agent i rewards in the *market*; with probability $p_i^{reward\_local}(t)$ the agent rewards within the *local group*. (The dynamics of the reward probabilities are again driven by various kinds of learning and will be explained later.)

3. Some *mobility* between groups may be possible. Different from the Macy-Sato-model[15] this mobility could be both, an endogenously given exchange of member and/or an exchange induced by meeting on the market. The latter would become more frequent as successfull market exchanges take place.

The PM-scenario has structural similarities with the Macy-Sato-model. An important *difference* is that we *differentiate* the probability for rewarding in the market from the probability for rewarding locally. That should allow analyzing conjectures and hypotheses of Macy and Sato in a similar – though much more reasonable – structural setting (and with – hopefully – improved components and modules for opportunity and transaction costs).[16]

Obviously, there are *further structural settings* to be analyzed in HUME$_{1.x}$. Good candidates would be network structures that – different from the GD-scenario above – are *not* grid based: The use of grids is a very *convenient*

[15]Michael Macy and Yoshimichi Sato, Trust, cooperation and market formation in the US and Japan. In: *Proceedings of the National Academy of Sciences*, 99, 2002, pp. 7214-7220.

[16]See Oliver Will and Rainer Hegselmann, On the computational model in "Michael W. Macy and Yoshimichi Sato: *Trust, Cooperation and Market Formation in the U.S. and Japan*. Proceedings of the National Academy of Sciences, May 2002". Submitted to JASSS.

modelling approach if one has to analyse or wants to model especially *local* interactions with *overlapping* neighbourhoods. *If* such a social topology – even as a stylised one – is untypical for the target system and possibly misleading because there are reasons to believe that the difference in the topological structure matters, *then* using grids would be rather dangerous: For instance, the very regular and overlapping neighbourhood structure has as a consequence an *extremely large diameter* of grid-based networks. However, most real word networks surprise by their small diameter. Therefore, it is a very natural idea to analyze a setting, in which – *similar* to the GD-scenario – network distance matters, but – different from the GD-scenario – the network itself is *not* grid-based. Another structural setting could consist of a kind of PM-scenario in which the local group has – different from the PM-scenario above – a relevant network structure that, for instance, affects learning, spread of information and reputation etc.

4 Detecting trustworthiness

Basically, the *ETG* has three moral dimensions: trustworthiness, honesty and reliability (as characterized in §2). All three cause serious detection problems. To simplify the situation we assume in HUME$_{1.0}$ reliability, while the problem of honesty-detection is resolved by an observability assumption. Thus, the only – though difficult enough – problem left, is the detection of trustworthiness, which is somehow hidden in the not directly observable character of agents, i.e. in their reward radius $r_i(t)$ or in their reward probabilities $p_i^{reward\_market}(t)$ and $p_i^{reward\_local}(t)$, respectively.

There are at least three starting points for a P-agent to form a belief about an S-agent's trustworthiness: reputation, signalling, and past experience.

1. *Reputation* is a kind of believe about the others' believes about somebody, a more or less inter-subjectively consistent more or less accurate social information and evaluation that spreads with a higher or lower speed in a population or network, respectively. Reputation can be build, get lost and even be managed or manipulated. (Example: Distorting others' reputation may increase a S-agent's chance to get the job.)

2. *Signalling* regards all the subtle, unintentionally given signals and clues that tell something about an agent's actual intentions, character traits and moral type.

3. *Past experience* with others – personal interactions or observations – allows to draw conclusions from their behaviour to their character traits. (However, the latter may have changed in the meantime.)

Reputation, signalling, and past experience can change over time and may point into different directions. The corresponding belief formation involves complex cognitive processes and capacities.

In HUME$_{1.0}$ we will try to cover and to incorporate these processes and capacities. However, we will do that by a *modelling short cut*: We simply *assume* that mechanisms like resorting to reputation, signalling, past experience etc. really work – though to a higher or lower degree. "*Really work*" means: As an assumed matter of fact an S-agent that is more (less) likely to reward (exploit) is more (less) likely to be classified as rewarder (exploiter)— and then treated accordingly.

The details for a PM-scenario: In that scenario we have the two *reward probabilities*, $p_i^{reward\_market}(t)$ and $p_i^{reward\_local}(t)$. Depending upon the reward probabilities of an S-agent, there is a certain probability – c_R^{market} and c_R^{local}, respectively – to be classified as a rewarder by P-agents. Correspondingly,

$$c_E^{market} = 1 - c_R^{market}$$

and

$$c_E^{local} = 1 - c_R^{local}$$

are the probabilities to be classified as an exploiter – either in the market or locally. The classification of an S-agent is carried out *individually* by P-agents and, therefore, can differ for different P-agents. As explained later, the classification has severe consequences for the matching procedure. (A classification as an exploiter excludes being matched with the classifying P-agent – though there may be another P-agent who did another classification). The classification functions should have the following properties:

1. They are *not* time-dependent: Detection does not become better over time – at least in HUME$_{1.0}$.

2. The functions are the same for all individuals: Compared to others, no one's detection capacities are better or worse – at least in HUME$_{1.0}$.

3. The classification functions *monotonically increase* for increasing values of $p_i^{reward\_market}(t)$ or $p_i^{reward\_local}(t)$: The more likely a S-agent rewards, the more likely he is classified as a rewarder by P-agents; the more likely

a S-agent exploits, the more likely he is classified as an exploiter by P-agents. (Corollary: $c_R^{market}$and c_R^{local} have a *minimum* for the reward-probability zero and a maximum for reward-probabilities of one.)

Infinitely many functions fulfil requirements (1) – (3). All these functions describe *macroscopically* (!) how good the combined effects of reputation, signalling, past experience etc. really work. HUME$_{1.0}$ will provide a framework in which certain functions and their effects can be analysed. Interesting functional scenarios can be defined by linear functions or by very simple linear approximations of *non*-linear functions. For the functions, one should, *firstly*, keep in mind that – since the y- and the x-axes represent probabilities – we are operating in the *unit square*. *Secondly*, whenever for the classification functions[17] it holds that[18] $c_R(p_i^{reward}(t)) > p_i^{reward}(t)$ then P-agents are *over-confident* with regard to the trustworthiness of S-agents: *Too often* they expect them to be trustworthy rewarders. Similarly, $c_R(p_i^{reward}(t)) < p_i^{reward}(t)$ implies *under-confidence* on the P-agents side. Only $c_R(p_i^{reward}(t)) = p_i^{reward}(t)$ is a *calibrated*, i.e. an appropriate classification, given the reward probabilities. With that in mind, interesting functional scenarios to start with in HUME$_{1.0}$ are, for instance, the following scenarios:

1. If being exploited is more costly than having no exchange at all, then circumstances may favour a bias in the direction of under-confidence. Thus, *both* functions, c_R^{market} and c_R^{local}, may express under-confidence, however, with different degrees. Intuitively it would be plausible to assume a much higher under-confidence for c_R^{market} than for c_R^{local}. Suitable variations of the parameters a and/or b in classification functions of the *shape* $f(x) = a \cdot x + b$ (the functions eventually 'cut off' to keep the values within the range $0 \leq f(x) \leq 1$) allow a systematic analysis of the resulting effects.

2. The most natural and simple functional start scenario is probably one in which we have two parts of a linearly approximated non-linear function. 'Naturally', though *not* necessarily so, the 'point of infliction' would be $\langle 0.5, 0.5 \rangle$. Then, over- and under-confidence and their respective degrees can be defined independently for both parts of both classification functions. That allows for scenarios in which, for instance, c_R^{market} and c_R^{local} are – except for .5 – underconfident in the left *and* the right part of the function, though to *different* degrees.

[17] As explained above, we assume this functions to be the same $\forall i, \forall t$.

[18] Since it is obvious what is meant, I do *not* differentiate here the reward probabilities in the different scenarios.

The way we model type classification so far, is *not without problems*: The classification mechanism described above reacts *too fast* on actual changes of the propensities to reward. Signalling may change immediately. But the change of reputation will have some *delay* and at least the recent past should always matter to a certain degree.– There are at least three ways to account for such effects.

1. The classification mechanism could operate on a *weighted* reward propensity defined as

$$p_i^{weighted\_reward}(t) = \alpha \cdot p_i^{reward}(t-1) + (1-\alpha)\, p_i^{reward}(t)$$

   In this convex combination α controls the weight of *past* reward propensities, while $(1-\alpha)$ is the weight of the *actual* propensity (which might affect actual signalling).

2. Another way to fix the delay problem could be to base the classification mechanism on the *real behavioural frequencies*

$$freq_i^{reward}(t_\theta)$$

   of rewarding behaviour in a shorter or longer past. The relevant past, i.e. the number of periods that are taken into account, is given by a parameter θ. The frequencies are defined as the ratio given by the number of rewarding behaviour divided by the number of occasions to reward or to exploit in the periods $(t-\theta)$ to t. To initialise simulation runs we could resort to the frequencies that correspond to the reward propensities in $t = 0$.

3. It makes sense to cross over the first and second approach: The classification mechanism operates on the actual reward propensities (affecting signalling) and the past reward frequencies (affecting reputation). As a result we get again a weighted reward propensity—now based on *real* behavioural frequencies rather than *expected* ones:

$$p_i^{weighted\_reward*}(t) = \alpha \cdot freq_i^{reward}(t_\theta) + (1-\alpha)\, p_i^{reward}(t)$$

Since it seems to be the most simple approach, we will start in HUME$_{1.0}$ with the *first* one.

The considerations above can be modified in such a way that they become applicable in a GD-scenario as well. The basic idea is that the probability for being classified as a rewarder depends on *two* variables: one's own (weighted)

reward probability *and* the network distance to the classifying other. A natural assumption *could* be, that the probability to be classified as a rewarder, firstly, monotonically *increases for increasing (weighted) reward probabilities*, but, secondly, monotonically *decreases with network distance*. As with regard to the PM-scenario, we assume the – now *two*-dimensional – function to be constant over time and the same for all agents.

For both, GD-scenario and the PM-scenario, HUME$_{1.0}$ will get a graphical user interface that allows for convenient experimentation. In the PM-context it should be possible to generate a linear approximation of a non-linear classification function with one serious[19] infliction point that can be chosen arbitrarily – simply by a mouse click in the unit square. In the GD-scenario it is a bit more complicated since the function is two-dimensional: Both dimensions are made discrete. Thereby a grid is generated. Mouse activities (click, double click) select grid points and allow entering values. Result is a *two*-dimensional '*classification landscape*'.

5 Matching P-agents and S-agents

In our setting – the GD-scenario or the PM-scenario – S-agents are of different attractiveness for a P-agent and *vice versa*. They have to find together by a *matching process*. The problem is a problem of *two-sided matching* (and not just one-sided-matching as for instance the *secretary problem* is). If matching agents are found, then they play the *ETG*.

Optimal matching exists! Given the problems of P-agents and given the competencies of S-agents both types of agents have rank ordered preferences over the possible partners of the opposite type. Under the assumption of equal sized numbers of agents in both sets, strict and complete preferences our matching problem is an instance of the *stable marriage problem* for which (type!)-optimal solutions exist. Optimality is defined in terms of stability: "A particular matching is *unstable* if there are two parties who are not matched with each other, each of whom strictly prefers the other to his/her partner in the matching. . . . A *stable* matching is. . . a matching that is not unstable."[20] Solution and stability concepts can be extended to work under weaker as-

[19] Others may result from cut off effects that keep the function within the range[0,1].

[20] Dan Gusfield / Robert W. Irving, *The stable marriage problem - Structure and Algorithms*, Cambridge (Mass.) 1989, MIT Press, page 2. See also: Alvine E. Roth / Marilda A. Oliveira Sotomayor, *Two-sided Matching - A Study in Game Theoretic Modeling and Analysis*, Cambridge 1990, Cambridge UP

sumptions (allowing indifferences, non equal sized sets etc.).

A first idea might be to resort in HUME$_{1.0}$ to the *Gale/Shapely-algorithm*[21] that computes an optimal solution—as one can prove. However, doing that comes down to assuming an *all-knowing central authority*, an *all-knowing social planner* who does the matching (as in case of the National Resident Matching Program – NRMP – that matches graduating medical students with hospitals in the US). Even extremely intelligent and very fast agents could never find the solution if their communication and knowledge is only local. Without an all-knowing central social planner, the matching generated by the *Gale/Shapely-algorithm can't even contra-factually be interpreted as the result of any 'feasible' decentralized social process* with only local and limited knowledge about others.

On the other side, it is tempting to try (again) a *modelling short cut* in order to avoid all the complicated details we get into once we start to model explicitly decentralized matching. There *is* such a short cut: We develop a matching procedure which at a certain point takes a social planner's perspective, *but* the planner is *not* all-knowing and *not* all-mighty rather than of a *boundedly rational type*. Thereby we generate a matching that boundedly rational agents – presumably and plausibly – *could have brought about by decentralized interactions*.– At least, that is the methodological hope accompanying the 'trick'.

The matching procedure is in the spirit of Herbert Simons' *satisficing* and borrows from the *Take-the-next-best-heuristics* as analyzed for mating processes by Todd and Miller.[22] Their basic idea is: "For some specified C, the first $C\%$ of the N total potential mates are checked (without being selected), and the highest dowry D is remembered – this is the researchers aspiration level. After the first $C\%$ of potential mates have gone by, the next potential mate with a dowry greater than D is chosen. (If no greater dowry turns up, then we assume that the searcher accepts the very last individual in the sequence, ...)" (Todd/Miller 295).

In HUME$_{1.0}$ the matching of P- and S-agents works as follows:

1. Every P- and every S-agent gets assigned a randomly generated subset

[21]For a description of the algorithm see Gusfield/Irving, Footnote 9

[22]Peter M. Todd / Geoffrey F. Miller, From Pride and Prejudice to Persuasion - Satisficing in Mate Search. In: Gerd Gigerenzer / Peter M. Todd and the ABC Group, *Simple Heuristics that Make Us Smart*, Oxford 1999, Oxford UP, chapter 13, 289-324.

of agents of the *other* type. The size of the subset is the same for all and exogenously determined either by absolute numbers or by fractions. The highest reward payoff one could get in an ETG played with an agent in one's subset defines the actual individual aspiration level: Period by period P-agents determine that level under the perspective of their actual problem k. As a consequence of this step, all agents that look for exchange partners have an aspiration level – either in the GD- or in the PM-scenario (some more scenario specific details will be given later).

2. Let m be the number of P-agents while n is the number of S-agents. We now calculate a $m \times n$ matrix that represents all matches that *mutually* would work – given the aspiration levels of both, the P-agent and the S-agent – and those that would not, either for both or for one of the two agents. Matches that work get a "+1" entry; those that don't work get a "–1"-entry. Let M_0 be the start matrix.

3. With probability $\frac{m}{m+n}$ we select a *P-type-perspective*, i.e. we will look along the *rows*; with probability $\frac{m}{m+n}$ we select a *S-type-perspective*, i.e. we will look along the *columns*. As will become clear soon, this perspective-lottery *avoids type privileges* in the matching procedure. Depending upon the result of the perspective lottery, then, with *equal* probabilities either one of the rows (P-type-perspective) or *one* of the columns (S-type-perspective) is randomly selected.

4. Suppose, the P-type-perspective and then a certain row was selected. (For the S-type-perspective and a certain column the procedure works analogously.) If there are "+1" entries in the *row, then* with equal probability one of them is randomly selected. If there are "–1" entries only, then with equal probability one of them is *randomly* selected. Thus, whenever there are positive entries, one of them – even if it is only one – *is* selected. Of course, a match with an S-agent that has a "–1"-entry makes sense only if such a pairing is for both agents better than having no partner at all—what is the case if the payoff for not being paired is set to 0. (However, one could think of alternatives in which the bias towards pairings with positive matrix entries is *less* pronounced.)

5. After a match according the steps (3) and (4) above is established, the involved row and column of the matching matrix M_0 are *eliminated*. The resulting new matrix M_1 is a $(m-1) \times (n-1)$ matrix.

6. On M_1 again the steps (2)–(5) are applied and so on[23] – until after an obviously *finite* number of row and column eliminations, we get a matrix which does not have any row or column.

The matching procedure according (1)–(6) has certain characteristics: Since the intersection of the P-agents and the S-agents is empty (due to the mechanism for the assignment of problems), nobody can get matched with himself. Thus, there is no *self-engagement*. Engagement is *exclusive* in the sense that everybody can get matched only once. Who, given the aspiration levels, is *more often mutually accepted* as a possible partner, has a *better chance to find a positive matching partner*. For the two types, P-agents and S-agents, there are *no one-sided type privileges* (as they exist – as gender privileges – in some versions of the *optimal* algorithms mentioned above): Both types have the same chance to find matching partners. There is *no guarantee that everybody is matched* – even if logically that would have been possible. Finally, there is no guarantee *that matches are stable* in the sense as defined above for the stable marriage problem. However, this effect is on purpose and an intended consequence: Basically we want to generate a matching as it plausibly *could have been brought about by decentralized interactions of agents that have only limited knowledge of the search pool and the agents therein*. Under such conditions, stability should be a rare event.

The procedure described above is applicable in both, the GD- and the PM-scenario. For the PM-scenario that is obvious. In the GD-scenario we have to consider *network distances*: The aspiration levels have to be specified based upon *local* samples of agents of the opposite type. 'Local' has to be defined in terms of neighbourhood/network distance. Given the aspiration levels the matrix M_0 can be calculated as described above. Then again the network distance matters: *Only* matches *within* the P-agents' search radius $radius_i^P(t)$ are possible. Matches that are not possible, therefore get a kind of out-of-range-entry, namely "0" in matrix M_0. After these modifications, the matching procedure works as described above.

Finally, we have to clarify the *interplay* of matching and detection. So far, detection results do not play any role in the matching procedure, which looks only at problems and competences. A natural idea that can easily be integrated in the matching algorithm is the following: All P-agents individually evaluate the trustworthiness of all S-agents and block all those they classify as exploiters according to the detection procedure described in §4.

[23] The m and n in step 3, then, is the m and n of the reduced matrix: $m \leftarrow m-1$ and $n \leftarrow n-1$.

For our matching algorithm that would mean: Between step (2) and (3) we introduce a new step – let's say the step (2a). In that step all S-agents j that are classified as exploiters by a P-agent i get a "0" entry at the matrix position $\langle i, j \rangle$. No S-agent that was classified by a P-agent as an exploiter, can be matched with that P-agent. As a consequence, there is a tendency that S-agents which tend to exploit, are *punished by exclusion*.

6 Excursion: Effect generating versus process representing modules

For both, the matching procedure and the detection processes, a *somehow indirect* modelling approach is used. With a focus on the *matching* procedure, one can say: The procedure described in §5 is not a procedure for agents and their activities (cognitive or social). What the procedure does is: *Directly generating* some effects which we, i.e. the *modellers*, think that agents – with all the limitations we want them to have – would or could bring about by their activities. However, these activities are not *explicitly modelled*. Of course, they could be modelled. In our case one could think of a certain range of vision agents have and that allows them to see some of the characteristics of others. In case of mutual vision, agents could address each other, make proposals and finally make a deal and so on. All these details are not *explicitly modelled* by the matching procedure above. Instead – as methodological trick or modelling short cut – a matching procedure that fits a boundedly rational social planner is used.

Not being explicit about the underlying processes is, trivially, a *disadvantage* if explicitness is the only goal. On the other hand, not being explicit and directly generating the intended effects makes it *easier* to control other parts the model – *if* one is sure about the intended effects. If in such a case, the explicit processes would not produce the directly generated effects, then, that would be a lack of adequacy.

Obviously, one can distinguish in models *two different types of modules*: **E**ffect-**G**enerating modules (EG-type) and **P**rocesses-**R**epresenting modules (PR-type). Models usually consist of a mix of both types of modules. In the process of modelling, corresponding types of modules can possibly be *checked against each other*. For instance, if one trusts more in the EG-module, then that module could be used to check the adequacy of a corresponding PR-module one is working on. However, such a check presupposes that both

modules and their relevant effects can be sufficiently separated, mapped and compared. Even more, it may be (and often is) impossible to check separately the explicit component in the complete and total model. At the same time, it may be a *non-trivial task* to make sure and demonstrate that an EG-module really produces all (and only) the effects it should produce. "*Black box*" is the term that is sometimes used to refer to modules we propose and would *prefer* to call *effect-generating module*. For instance, the matching procedure above is not a process-representing module. Nevertheless, one *cannot* say that the pairing mechanism above is a *black box* since everything going on in the module is transparent, clear, and even 'explicit' in a certain sense.

As a matter of fact, EG-modules are often much more simple. In such a case a more simple module / model can become a good test bed to study advantages and disadvantages of a more explicit approach to agents' activities. It is not *necessarily* so that a module that is not explicit with regard to agents' actions and processes, is *the more simple module* in terms of length of the involved algorithms, running time, required memory or other computational resources. In principle, functionally equivalent EG- and PR-modules could behave very different with regard to computational complexity. Given their functional equivalence one of them could be a much faster, cheaper or easier to handle solution.

It might be a good idea, that even on the level of flow charts and pseudo code, EG-modules are marked as such and, at the same time, get assigned an accompanying **P**lausible **N**arratives (PN) – again, marked as such symbolically – that tells a plausible and well informed informal short story about the processes that probably are or might be involved. Of course, somehow almost all modules need some informal interpretation and explanation. However, the point is, that for an EG-module it is *part* of the accompanying PN that the EG-module "*is not the true story*", i.e. does *not* represent the real processes at work—not even in a stylized form and under a fairly liberal understanding of 'representation'.

7 Learning, communication, and character transformation

The agents' characters change over time. By working on problems they develop special *technical* craft skills – a process represented and modelled by the dynamical competence vector. *Moral* character traits of agents change

as well. They develop, evolve or erode by learning and / or communication.

How the the agents' moral character traits are modelled, is scenario-dependent: In the GD-scenario *distance* is decisive. Agents with problems, i.e. the P-agents, search for good S-agents only within a certain radius. S-agents reward *within* a certain radius with a certain propensity, that is different from the one *beyond* that radius. Thus, we can gather an agent's i decisive characteristics in a 'morals vector' $M_i^{GD}(t)$. The time dependent vector has a *four* components and the following structure:

$$M_i^{GD}(t) = \left\langle radius_i^P(t),\ radius_i^S(t),\ p_i^{reward \leq radius}(t),\ p_i^{reward > radius}(t) \right\rangle$$

The corresponding morals vector for the PM-scenario is somewhat different. Not distance, rather than *location* matters—either the local group or the market. An agent i has as P-agent and as S-agent a *specific* propensity to stay local or to enter the market. Then, in an S-agent's role, an agent has location dependent propensities to reward. The structure of the vector – again time dependent and consisting of four components – is given by

$$M_i^{PM}(t) = \left\langle p_i^{P \to market}(t),\ p_i^{S \to market}(t),\ p_i^{reward\_local}(t),\ p_i^{reward\_market}(t) \right\rangle$$

Figure 10 gives an overview and comparison.

The transformation of an agent's morals is always *success driven*: Agents may learn this way or that way, nevertheless, all learning mechanism are such that trusting and trustworthiness *erodes* as morality tends to become a *'looser strategy'*. Interesting transformation mechanisms are at least the following:

1. *Discussions* with others: The basic idea is that from time to time agents meet and discuss the right propensities as they are relevant in the different scenarios. The meetings (at the end of a long day, with a coconut milk or a Campari soda – to come up with a hint to plausible narratives) may take place in the market or in one's local group, within one's close neighbourhood or in a more distant part of the grid; the meetings may involve more or less agents. To model the discussions we can resort to *models of opinion dynamics*. Opinions are certain behavioural propensities, for instance, the propensity to reward in the market. It would be very natural to use weights that reflect success in terms of relative payoffs (present or over a longer or shorter past) and, then, to update opinions, i.e. propensities, by *weighted averaging* over

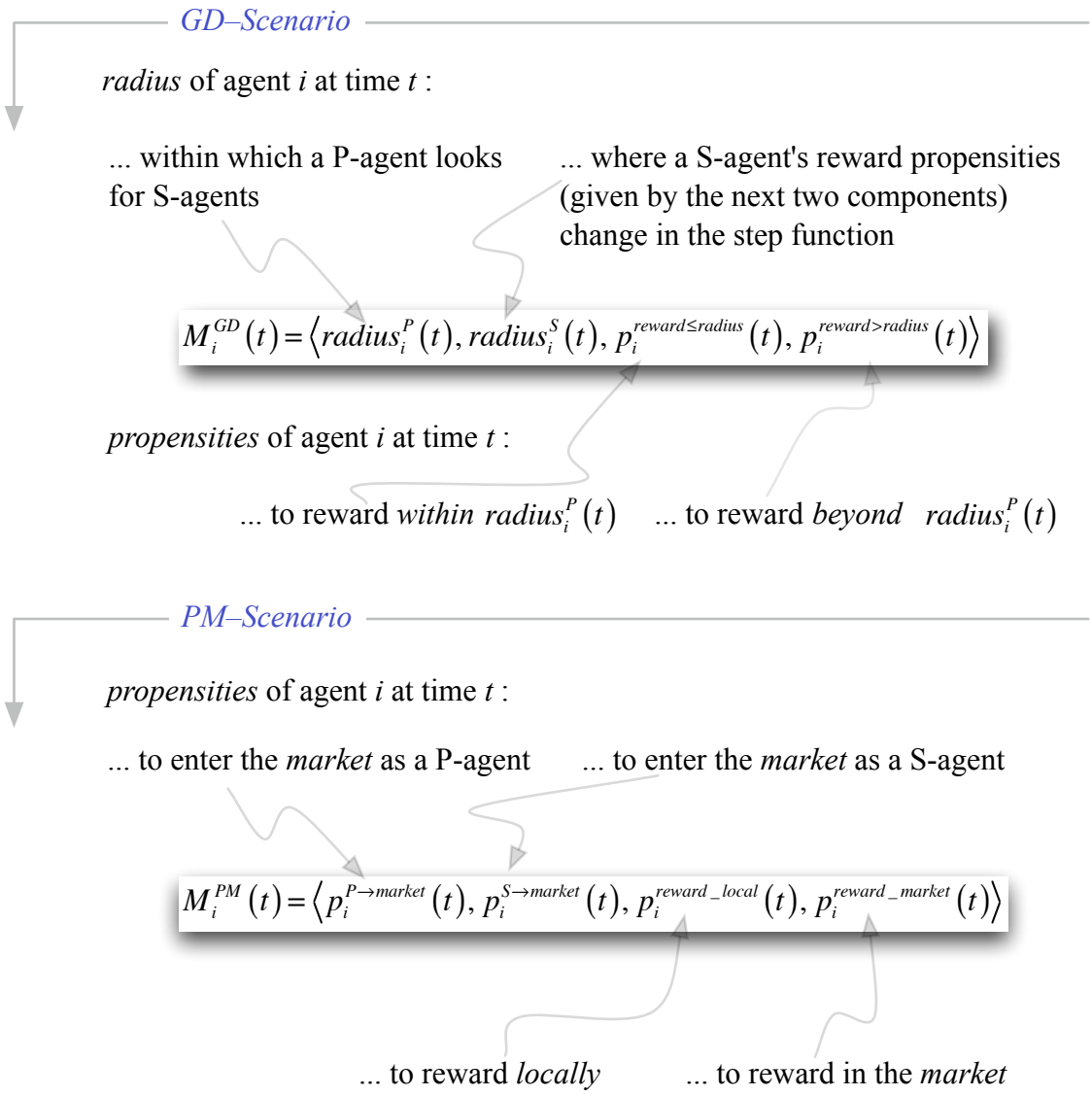

Figure 10: Morals vector for both scenarios

some rounds of a stylised discussion. This type of opinion dynamics is known as the *Lehrer/Wagner-model*[24].

A *serious problem* is that such an approach assumes *truthfulness* as to one's propensities in a context where agents which are likely to exploit, should have a strong incentive to keep that fact secret. A way out might be to restrict stylised discussions to sets of agents, that do not tend to exploit one another, as for instance next neighbours in the GD-scenario. Another (complicated) strategy could be to build in more or less reliable truthfulness detection *and* cheating capabilities.

2. Evaluation of *one's own* payoff-experience (reinforcement learning): TO BE ELABORATED.

3. *Social comparisons* and *imitation* of more successful others: Within a certain *learning neighbourhood* (GD-scenario) or within their local group (PM-scenario) agents are randomly matched. The agents can mutually *observe* both, success in terms of payoffs *and* all the entries in the morals vectors. In the GD-scenario the learning neighbourhood is measured in terms of network distance, is given by a corresponding parameter. Social comparisons could be based on the last period only, but it is probably better if the past – at least the recent past – would count as well. How to account for the past? There are at least one easy to handle approach: A certain number of periods counts. The social comparison is based on average values – payoffs, propensities, and radii – over this number of periods.

   To draft a comparison mechanism is easier for the PM-scenario, since there come into play *only* propensities rather than propensities *and* radii. For every agent we need payoff information for the relevant past. This information would have the components:

   (a) payoffs from going to the market as S-agent
   (b) payoffs from going to the market as P-agent and rewarding there
   (c) payoffs from going to the market as P-agent and exploiting there
   (d) payoffs from staying local as P-agent and rewarding there
   (e) payoffs from staying local as P-agent and exploiting there

[24]K. Lehrer / K. Wagner, *Rational consensus in science and politics*, Dordrecht: D. Reidel Publ. Co, 1981. An overview on models of opinion dynamics can be found in R. Hegselmann / Ulrich Krause, *Opinion Dynamics and Bounded Confidence – Models, Analysis, and Simulations*: Journal of Artificial Societies and Social Simulation (JASSS) vol.5, 2002, no. 3 http://www.soc.surrey.ac.uk/JASSS/5/3/2.html.

Learning than would mean: Adaptation of one's propensities in the direction of the other's propensities if and only the other one's corresponding payoffs are higher than one's own payoffs. ELABORATION NECESSARY. ICH BIN EINFACH SCHRECKLICH LANGSAM.

Note: Social comparison and imitation as explained here, seem to work into the same direction as discussions *without* deceit.

4. *Mutation* There is always a small probability that a component of the morals vector is changed by certain random amount not greater than a certain value—probability and upper limit given as parameters of the model that experimenters can manipulate.

At the beginning – i.e. in HUME$_{1.0}$ – only the last two mechanism will be analysed. In a first step they will be analysed *separately*: One and only one mechanism is at work in a simulation run. Later – in HUME$_{1.x}$ – we will assume a mix of mechanisms or populations that are heterogeneous with regard to their transformation policies.