5.6 Knowledge gain

Knowledge and communication are closely related in a.m.p. systems. For an event e and a run r write $e \in r(m)$ when there is a process i such that $e \in r_i(m)$ and write $e \in r$ when there is a time m such that $e \in r(m)$. Furthermore, write $e \in r_i$ when there is a time m such that $e \in r_i(m)$.

Definition 5.6. Let $i_1, \ldots, i_k$ be a sequence of processes, not necessarily distinct, let r be a run, and let $m < m'$. Then,

$$\langle i_1, \ldots, i_k \rangle$$

is a process-chain in $(r, m..m')$ if there are events $e_1, \ldots, e_k$ in run r such that

$$\begin{aligned} &e_1 \notin r(m) \\ &e_k \in r(m') \\ &e_j \in r_{i_j} \end{aligned}$$

for $j = 1, \ldots, k$, and

$$e_1 \xrightarrow{r} \cdots \xrightarrow{r} e_k$$

Example 5.7. Let r be a run. Suppose process 1 sends a message to process 2 in round 1, suppose process 2 receives the message in round 2 and sends a new message back to process 1 in round 3, which process 1 receives in the same round. Then, $\langle 1, 2, 2, 1 \rangle$ and $\langle 1, 2, 1 \rangle$ are process-chains in $r(0..3)$.

We say that $\langle i_1, \ldots, i_k \rangle$ is a process-chain in r if there are times m and m' such that $\langle i_1, \ldots, i_k \rangle$ is a process-chain in $r(m..m')$.

Lemma 5.8. *Assume that $\langle i_1, \ldots, i_k \rangle$ is a process-chain in $r(m..m')$ such that $i_j \neq i_{j+1}$ for $j = 1, \ldots, k-1$. Then, there is a sequence of messages $\mu_1, \ldots, \mu_{k-1}$ such that $send(\mu_1, \cdot, i_1) \notin r(m)$, $send(\mu_{j-1}, \cdot, i_{j-1}) \notin r(m'')$ and $send(\mu_{j-1}, \cdot, i_{j-1}) \in r(m''+1)$ implies $send(\mu_j, \cdot, i_j) \notin r(m''+1)$, and $send(\mu_j, \cdot, i_j) \in r(m')$. In particular, at least $k-1$ messages must be sent between rounds $m+1$ and m' inclusive.*

Proof. Since $\langle i_1, \ldots, i_k \rangle$ is a process-chain in $(r, m..m')$, there are events $e_1, \ldots, e_k$ in run r such that $e_1 \notin r(m)$, $e_k \in r(m')$, $e_j \in r_{i_j}$ for $j = 1, \ldots, k$, and $e_1 \xrightarrow{r} \cdots \xrightarrow{r} e_k$.

Let m_j denote the round in which e_j occurs. As $e_j \xrightarrow{r} e_{j+1}$, $e_j \in r_{i_j}$, $e_{j+1} \in r_{i_{j+1}}$, and $i_j \neq i_{j+1}$, there is a message μ_j such that $send(\mu_j, \cdot, i_j) \in r_{i_j}$ and

$$e_j \xrightarrow{r} send(\mu_j, \cdot, i_j) \xrightarrow{r} e_{j+1}$$

Hence, $send(\mu_j, \cdot, i_j) \in r_{i_j}(m'')$ for $m_j \le m'' \le m_{j+1}$. Moreover,

$$send(\mu_j, \cdot, i_j) \xrightarrow{r} send(\mu_{j+1}, \cdot, i_{j+1})$$

implies that the two send events can not occur in the same round. $\square$

For a sequence of processes $i_1, \dots, i_k$, write $(r, m) \sim_{i_1,\dots,i_k} (r', m')$ if there are points $(r_0, m_0), \dots, (r_k, m_k)$ such that

$$(r, m) = (r_0, m_0) \sim_{i_1} (r_1, m_1) \sim_{i_2} \cdots \sim_{i_k} (r_k, m_k) = (r', m')$$

When this holds we say that (r', m') is $(i_1, \dots, i_k)$-reachable from (r, m).

Lemma 5.9. *Let $\mathcal{R}$ be an a.m.p. system in which each process performs at most one action per round, let $r \in \mathcal{R}$, and let $m < m'$. Then, for all $k > 0$ and for all sequences of processes $i_1, \dots, i_k$, either*

$$(r, m) \sim_{i_1,\dots,i_k} (r, m')$$

or $\langle i_1, \dots, i_k \rangle$ is a process-chain in $(r, m..m')$.

Proof. By induction on k. If $k = 1$ and $\langle i_1 \rangle$ is not a process-chain in $(r, m..m')$, then there is no event e such that $e \notin r_{i_1}(m)$ and $e \in r_{i_1}(m')$. Hence, $r_{i_1}(m) = r_{i_1}(m')$ and $(r, m) \sim_{i_1} (r, m')$ as desired.

Suppose $k > 1$ and take as induction hypothesis that for all sequences of processes $i_1, \dots, i_{k-1}$ and runs r, either

$$(r, m) \sim_{i_1,\dots,i_{k-1}} (r', m')$$

or $\langle i_1, \dots, i_{k-1} \rangle$ is a process-chain in $(r, m..m')$.

Assume $\langle i_1, \dots, i_k \rangle$ is not a process-chain in $(r, m..m')$. Let e^* be the last event in the history of i_k at (r, m'). Define a new run r' as follows

$$r'_i(m'') = \begin{cases} r_i(m'') & \text{if } m'' \le m, \\ r'_i(m''-1) \cdot \langle \{e \in r_i(m'') \mid e \xrightarrow{r} e^*, & \\ \qquad\qquad\qquad\qquad e \notin r'_i(m''-1)\} \rangle & \text{if } m < m'' \le m', \\ r'_i(m') & \text{if } m'' > m'. \end{cases}$$

Then, $r'(m'')$ is a prefix of $r(m'')$ for every $m'' \ge 0$ since $e \xrightarrow{r} e^*$, $e' \in r_i(m'')$, and $e' \xrightarrow{r} e$ implies $e' \xrightarrow{r} e^*$. Moreover, r' satisfies MP1–MP3.

MP1: $r'_i(m'')$ is a history since it is a prefix of $r_i(m'')$ which in turn is a history.

MP2: If the event $receive(\mu, j, i)$ is in $r'_i(m'')$, then it is in $r_i(m'')$ which implies that there is a corresponding event $send(\mu, i, j) \in r_j(m'')$. But as

$receive(\mu, j, i) \xrightarrow{r} e^*$ and $send(\mu, i, j) \xrightarrow{r} receive(\mu, j, i)$, $send(\mu, i, j) \xrightarrow{r} e^*$ and therefore $send(\mu, i, j) \in r'_j(m'')$.

MP3: $r'_i(0)$ is a sequence of length one as $r_i(0) = r'_i(0)$. The local state $r'_i(m''+1)$ is by construction either identical to $r_i(m'')$ or the result of appending a set of events to $r'_i(m'')$.

If $e \in r_{i_k}(m'')$ for $m < m'' \leq m'$, then $e \xrightarrow{r} e^*$ as $e^* \in r_{i_k}(m')$. Therefore, $e \in r'_{i_k}(m'')$ and $r'_{i_k}(m') = r_{i_k}(m')$ which implies $(r', m') \sim_{i_k} (r, m')$. Since $r(m) = r'(m)$, we trivially have $(r, m) \sim_{i_1} (r', m)$.

Assume that $\langle i_1, \ldots, i_{k-1} \rangle$ is a process-chain in $(r', m..m')$ and therefore also in $(r, m..m')$. Then, there exist events $e_1, \ldots, e_{k-1}$ in r' such that $e_1 \notin r'(m)$, $e_k \in r'(m')$, $e_j \in r'_{i_j}$ for $j = 1, \ldots, k-1$, and $e_1 \xrightarrow{r} \cdots \xrightarrow{r} e_{k-1}$. Since $e^* \in r_{i_k}(m')$ and since $e_{k-1} \xrightarrow{r} e^*$, $\langle i_1, \ldots, i_k \rangle$ is a process chain in $(r, m..m')$. This contradicts the induction hypothesis. Thus, $\langle i_1, \ldots, i_{k-1} \rangle$ is not a process chain in $(r', m..m')$ and consequently $(r', m) \sim_{i_1, \ldots, i_{k-1}} (r', m')$ from which follows $(r, m) \sim_{i_1, \ldots, i_k} (r, m')$. □

Theorem 5.10. *Let r be a run in an interpreted a.m.p. system $\mathcal{I}$ in which each process performs at most one action per round, and assume that $m < m'$.*

1. *If $(\mathcal{I}, r, m) \models \neg K_{i_k}\varphi$ and $(\mathcal{I}, r, m') \models K_{i_1} \cdots K_{i_k}\varphi$, then $\langle i_k, \ldots, i_1 \rangle$ is a process-chain in $(r, m..m')$.*
2. *If $(\mathcal{I}, r, m) \models K_{i_1} \cdots K_{i_k}\varphi$ and $(\mathcal{I}, r, m') \models \neg K_{i_k}\varphi$, then $\langle i_1, \ldots, i_k \rangle$ is a process-chain in $(r, m..m')$.*

Proof. 1. Assume that $\langle i_k, \ldots, i_1 \rangle$ is not a process-chain in $(r, m..m')$. Then, $(r, m) \sim_{i_k, \ldots, i_1} (r, m')$ by Lemma 5.9 and there are points in $\mathcal{I}$ such that

$$(r, m) = (r_0, m_0) \sim_{i_k} (r_1, m_1) \sim_{i_{k-1}} \cdots \sim_{i_1} (r_k, m_k) = (r, m')$$

If $(\mathcal{I}, r, m') \models K_{i_1} \cdots K_{i_k}\varphi$, then $(\mathcal{I}, r_j, m_j) \models K_{i_{k-j+1}} \cdots K_{i_k}\varphi$ for $j = 1, \ldots, k$. Hence, $(\mathcal{I}, r_1, m_1) \models K_{i_k}\varphi$ and $(\mathcal{I}, r', m') \models \varphi$ for all $(r', m') \sim_{i_k} (r_1, m_1) \sim_{i_k} (r, m)$. Consequently, $(\mathcal{I}, r, m) \models K_{i_k}\varphi$.

2. Proved similarly. □

Thus, a process can only gain or lose knowledge by sending messages. As a direct consequence common knowledge can neither be gained nor lost in a.m.p. systems.

Theorem 5.11. *Suppose r is a run in an interpreted a.m.p. system $\mathcal{I}$ in which each process performs at most one action per round, and suppose G is a group of processes such that $|G| \geq 2$. Then, for all formulas φ and all times $m \geq 0$*

$$(\mathcal{I}, r, m) \models C_G\varphi \quad \text{iff} \quad (\mathcal{I}, r, 0) \models C_G\varphi$$

Proof. Assume $(\mathcal{I}, r, m) \models C_G\varphi$ and $(\mathcal{I}, r, 0) \models \neg C_G\varphi$. Furthermore, assume that exactly l messages are sent during the rounds 1 to m. As $(\mathcal{I}, r, 0) \models \neg C_G\varphi$, there is a sequence $i_1, \ldots, i_k$ of pairwise distinct processes in G such that $(\mathcal{I}, r, 0) \models \neg K_{i_k} \cdots K_{i_1}\varphi$ ($\sim_i$ is an equivalence relation).

Let i and j be two distinct processes in G such that $j \neq i_k$. Since $(\mathcal{I}, r, m) \models C_G\varphi$, $(\mathcal{I}, r, m) \models (K_iK_j)^l K_{i_k} \cdots K_{i_1}\varphi$. By Theorem 5.10(1), $\langle i_k, j, i, \ldots, j, i\rangle$ is a process-chain in $(r, 0..m)$. Now, Lemma 5.8 implies that at least $2l$ messages are sent in r during the rounds 1 to m. This is a contradiction.

The case $(\mathcal{I}, r, m) \models \neg C_G\varphi$ and $(\mathcal{I}, r, 0) \models C_G\varphi$ is proved in the same way. □

6 Protocols and programs

We formalize the notion of actions taken by agents in multi-agent systems. When an action is taken by an agent, the local state of the agent changes. Let ACT_e and ACT_i be the set of actions for respectively the environment and the agent i. Furthermore, assume there is a null action λ corresponding to the environment or agent not performing any action. Since the actions of the agents may interact as they change the global state, define a joint action to be a tuple

$$(a_e, a_1, \ldots, a_n)$$

where $a_e \in ACT_e$ and $a_i \in ACT_i$. Let a global state transformer $\mathcal{T} : \mathcal{G} \mapsto \mathcal{G}$ describe a change of the global state of a system that corresponds to a joint action. Then, for every joint action $(a_e, a_1, \ldots, a_n)$ there is a global state transformer $\tau(a_e, a_1, \ldots, a_n)$ given by the transition function

$$\tau : ACT_e \times ACT_1 \times \cdots \times ACT_n \mapsto (\mathcal{G} \mapsto \mathcal{G})$$

Example 6.1. We return to the bit-transmission problem. The sender S either sends a bit or does nothing. Therefore, we can take the action of S to be $ACT_S = \{sendbit, \lambda\}$. Similarly, we set $ACT_R = \{sendack, \lambda\}$. The environment determines whether the message is delivered in the same round or lost. Thus,

$$ACT_e = \{(\lambda_S, \lambda_R), (deliver_S, \lambda_R), (\lambda_S, deliver_R), (deliver_S, deliver_R)\}$$

where $deliver_S$ and $deliver_R$ stands for delivery of the message sent by respectively S or R, and λ_S and λ_R corresponds to the environment losing the messages. Then, the transition function τ can be defined as follows.

$$\begin{aligned}
\tau\big((\lambda_S, \lambda_R), a_S, a_R\big)(s_e, s_S, s_R) &= \big(s_e \cdot (a_S, a_R), s_S, s_R\big) \\
\tau\big((\mathit{deliver}_S, \lambda_R), a_S, a_R\big)(s_e, s_S, s_R) &= \big(s_e \cdot (a_S, a_R), s_S, k\big) \\
\tau\big((\lambda_S, \mathit{deliver}_R), a_S, a_R\big)(s_e, s_S, s_R) &= \big(s_e \cdot (a_S, a_R), (k, \mathit{ack}), s_R\big) \\
\tau\big((\mathit{deliver}_S, \mathit{deliver}_R), a_S, a_R\big)(s_e, s_S, s_R) &= \big(s_e \cdot (a_S, a_R), (k, \mathit{ack}), s_R\big)
\end{aligned}$$

where $s_S = k$ or $s_S = (k, \mathit{ack})$ for $k = 0, 1$. Note that the transition function only makes sense for the states and actions that are valid in $\mathcal{R}^{bt}$.

Example 6.2. In an a.m.p. system, where only one action per process and round takes place, we can let the environment decide if and when a message is delivered. For example, assume that ACT_e consists of actions a_e of the form $\langle a_{e_1}, \ldots, a_{e_n} \rangle$, where a_{e_i} is either $\mathit{deliver}_i(\mathit{current}, j)$, $\mathit{deliver}_i(\mu, j)$, go_i, or nogo_i standing for immediate delivery of the current message from j to i, delivery of the message μ from j to i, performance of an action by process i, and idleness of process i. Let the set ACT_i consist of actions of the form $\mathit{send}(\mu, j)$ and of the internal actions in INT_i.

Suppose that $\tau(a_e, a_1, \ldots, a_n)(s_e, s_1, \ldots, s_n) = (s'_e, s'_1, \ldots, s'_n)$ and suppose that $a_e = \langle a_{e_1}, \ldots, a_{e_n} \rangle$. Then,

- s'_e is the result of appending $\langle a_{e_1}, \ldots, a_{e_n} \rangle$ to s_e,
- if $a_{e_i} = \mathit{go}_i$ and a_i is an internal action a or a send action $\mathit{send}(\mu, j)$, then s'_i is the result of appending the event $\mathit{int}(a, i)$ or $\mathit{send}(\mu, j, i)$ to s_i,
- if $a_{e_i} = \mathit{deliver}_i(\mathit{current}, j)$, $a_{e_j} = \mathit{go}_j$, and $a_j = \mathit{send}(\mu, i)$, then s'_i is the result of appending $\mathit{receive}(\mu, j, i)$ to s_i,
- if $a_{e_i} = \mathit{deliver}_i(\mu, j)$, then s'_i is the result of appending $\mathit{receive}(\mu, j, i)$ to s_i, and
- in all other cases $s'_i = s_i$.

6.1 Protocols

A protocol for a process i is a function

$$P_i : L_i \mapsto 2^{ACT_i} - \emptyset$$

that maps its local states to nonempty subsets of its actions. A set of actions is therefore used to denote the possible actions a process can take at a given local state. Thus, a protocol may be nondeterministic. We say that P_i is deterministic if $|P_i(s_i)| = 1$ for all $s_i \in L_i$, and write $P_i(s_i) = a$ rather than $P_i(s_i) = \{a\}$ in this case. Typically, the protocols of the processes are deterministic while the protocol of the environment is not.

A joint protocol P is a tuple $(P_1, \ldots, P_n)$ consisting of the protocols of the agents in the system and it describes the behavior of the agents. In order to be able to describe the behavior of the whole multi-agent system, we also need the protocol of the environment P_e, the set of possible initial global states $\mathcal{G}_0$, the transition function τ, and the set of acceptable runs Ψ. We gather this information into a tuple $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$ called the context.

In the bit-transmission problem and in a.m.p. systems we assume that the environment keeps track of the sequence of joint actions that are performed. This can be formalized as a context. Call $(P_e, \mathcal{G}_0, \tau, \Psi)$ a recording context if

- the state of the environment contains a distinct part h that is a sequence of joint actions,
- in all global initial states in $\mathcal{G}_0$, the state of the environment has $h = \langle\rangle$ the empty sequence of joint actions, and
- if $\tau(a_e, a_1, \ldots, a_n)(s_e, s_1, \ldots, s_n) = (s'_e, s'_1, \ldots, s'_n)$, then the sequence of joint actions h' in s'_e is the result of appending $(a_e, a_1, \ldots, a_n)$ to the h of s_e.

Example 6.3. Consider a message-passing system over the initial states Σ_i, the internal actions INT_i, and the messages MSG. A context $(P_e, \mathcal{G}_0, \tau, \Psi)$ is a message-passing context if

- the actions of process i are sets of elements of the form $send(\mu, j)$ for $\mu \in MSG$ and a for $a \in INT_i$,
- the local states of the processes are histories,
- for every initial global state $(s_e, s_1, \ldots, s_n) \in \mathcal{G}_0$, $s_i \in \Sigma_i$ for $i = 1, \ldots, n$, and
- if $\tau(a_e, a_1, \ldots, a_n)(s_e, s_1, \ldots, s_n) = (s'_e, s'_1, \ldots, s'_n)$, then either $s'_i = s_i$ or s'_i is the result of appending to s_i the set of events that have occurred for $i = 1, \ldots, n$ in the most recent round.

An interpreted context is a pair (γ, π) consisting of a context γ and an interpretation π.

Definition 6.4. A run r is consistent with a joint protocol $P = (P_1, \ldots, P_n)$ in a context $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$ if

- $r(0) \in \mathcal{G}_0$,
- for all $m \geq 0$, if $r(m) = (s_e, s_1, \ldots, s_n)$ then there is a joint action $(a_e, a_1, \ldots, a_n) \in P_e(s_e) \times P_1(s_1) \times \cdots \times P_n(s_n)$ such that $r(m+1) = \tau(a_e, a_1, \ldots, a_n)\big(r(m)\big)$, and
- $r \in \Psi$.

Hence, r is consistent with P in γ if r is a possible behavior of the system that follows the protocol P and is allowed according to the context γ. We say that r is weakly consistent with P in γ if r satisfies the first two conditions, i.e., if r is consistent with the step by step behavior of P in γ.

It is possible that no run is consistent with P in γ. In this case P is inconsistent with γ, otherwise P is consistent with γ. The protocol P is inconsistent with γ precisely when there are no runs in Ψ that are weakly consistent with P in γ. Generalizing, we say that a system $\mathcal{R}$ or an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ is consistent with P in γ if every run in $\mathcal{R}$ is consistent with P in γ.

Let the system representing P in γ, $R^{rep}(P, \gamma)$, be the system consisting of all runs consistent with P in γ. Similarly, let the interpreted system representing P in γ be $I^{rep}(P, \gamma, \pi) = \big(R^{rep}(P, \gamma), \pi\big)$. Notice that a system $\mathcal{R}$ is consistent with P in γ if and only if $\mathcal{R} \subseteq R^{rep}(P, \gamma)$.

Definition 6.5. A protocol P'_e is a restriction of P_e, written $P'_e \subseteq P_e$, if $P'_e(s_e) \subseteq P_e(s_e)$ for every local state $s_e \in L_e$. A context $\gamma' = (P'_e, \mathcal{G}'_0, \tau, \Psi')$ is a sub-context of $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$, written $\gamma' \subseteq \gamma$, if $P'_e \subseteq P_e$, $\mathcal{G}'_0 \subseteq \mathcal{G}_0$, and $\Psi' \subseteq \Psi$. Similarly, an interpreted context (γ', π) is a sub-context of (γ, π) if $\gamma' \subseteq \gamma$.

Lemma 6.6. *$\mathcal{R}$ is consistent with P in γ if and only if $\mathcal{R} = R^{rep}(P, \gamma')$ for some $\gamma' \subseteq \gamma$.*

Proof. If $\mathcal{R} = R^{rep}(P, \gamma')$ for some $\gamma' \subseteq \gamma$, then it immediately follows by the definition of consistency that $\mathcal{R}$ is consistent with P in γ. Conversely, if $\mathcal{R}$ is consistent with P in γ, then $\mathcal{R} = R^{rep}(P, \gamma')$ for $\gamma' = (P_e, \mathcal{G}_0, \tau, \mathcal{R})$ which is a sub-context of γ. □

Example 6.7. We describe the protocols and context of the bit transmission problem. The protocol P^{bt}_S of the sender S is defined by

$$P^{bt}_S(0) = P^{bt}_S(1) = \mathit{sendbit}$$
$$P^{bt}_S(0, \mathit{ack}) = P^{bt}_S(1, \mathit{ack}) = \lambda$$

and the protocol P_R^{bt} of the receiver R is

$$P_R^{bt}(\lambda) = \lambda$$
$$P_R^{bt}(0) = P_R^{bt}(1) = sendack$$

The context for the joint protocol $P^{bt} = (P_S^{bt}, P_R^{bt})$ is $\gamma^{bt} = (P_e^{bt}, \mathcal{G}_0, \tau, true)$, where

$$P_e^{bt}(s_e) = ACT_e$$
$$\mathcal{G}_0 = \{(\langle\rangle, 0, \lambda), (\langle\rangle, 1, \lambda)\}$$

and ACT_e and τ are as described in Example 6.1, and $true$ stands for all runs.

6.2 Programs

We use programs to describe protocols. A standard program for agent i is a statement of the form

case of
  if t_1 **do** a_1
  if t_2 **do** a_2
  $\vdots$
end case

where the t_j's are standard tests for agent i and $a_j \in INT_i$ (if the program contains only one if clause, we omit the case statement.) A standard test is a propositional formula such that its truth value depends only on the local state of i. An interpretation π on the global states $\mathcal{G}$ is compatible with a program Pg_i for agent i if every proposition that appears in Pg_i is local to i, i.e., if $s \sim_i s'$ for $s, s' \in \mathcal{G}$ implies $\pi(s)(p) = \pi(s')(p)$ for every proposition p that appears in a test in Pg_i.

If φ is a propositional formula whose primitive propositions are local to i, then we write $(\pi, s_i) \models \varphi$ if φ is satisfied by the truth assignment $\pi(s)$ for a global state $s = (s_e, s_1, \ldots, s_n)$. Given a program Pg_i and an interpretation π compatible with Pg_i, we let the program define a protocol

$$Pg_i^\pi(s_i) = \begin{cases} \{a_j \mid (\pi, s_i) \models t_j\} & \text{if there is a } j \text{ such that } (\pi, s_i) \models t_j \\ \{\lambda\} & \text{otherwise} \end{cases}$$

Hence, a program can be seen as a protocol, and we consequently generalize every notion regarding protocols to programs.

Example 6.8. In the bit-transmission problem the sender runs the program BT_S

$$\textbf{if } \neg recack \textbf{ do } sendbit$$

and the receiver runs the program BT_R

$$\textbf{if } recbit \textbf{ do } sendack$$

Let $BT = (BT_S, BT_R)$. Then, π^{bt} is compatible with BT. In fact, $BT^{\pi^{bt}} = P^{bt}$.

A specification is a desired property of a multi-agent system and therefore corresponds to a class of interpreted systems. An interpreted system $\mathcal{I}$ is said to satisfy a specification σ if $\mathcal{I} \in \sigma$. A run-based specification is a specification that is given as a property of runs. For instance, the property "the receiver eventually receives the bit from the sender" is a run-based specification that can be expressed as $\Diamond recbit$. Specifications that depend on the knowledge of the agents are called knowledge-based. For example, the specification "A child says 'Yes' if he knows that he is muddy" is a knowledge-based specification.

Definition 6.9. A protocol P satisfies a specification σ in the interpreted context (γ, π) if $I^{rep}(P, \gamma, \pi) \in \sigma$. A protocol P strongly satisfies σ in (γ, π) if every interpreted system consistent with P in (γ, π) satisfies σ.