

Algorithmic Analysis of Information Flow

Ron van der Meyden
University of New South Wales &
National ICT Australia

Information Flow Security

Example: a multi-user operating system, shared by both High (top secret) and Low (public information) level users.

A *covert channel* is an unintended way for High information to flow to Low.

Example: High and Low share a peripheral P , but have an ability to lock it for their own exclusive use.

High can send a sequence of bits b_0, b_1, \dots to Low:

during interval $[100k, 100k + 25]$: P locked by High iff $b_k = 1$

Logic of Knowledge and Linear Time

A propositional modal logic with constructs:

1. $K_i\varphi$ — agent i knows that φ
2. $\bigcirc\varphi$ — φ at the next moment of time
3. $\varphi_1 U \varphi_2$ — φ_1 holds until φ_2 becomes true
4. $\diamond\varphi$ — eventually φ holds ($= \text{true} U \varphi$)

Formulas combine these operators:

$$K_1 \diamond K_2 p$$

Environments

A (*finite*) *interpreted environment* for n agents is a tuple E of the form $\langle S_e, I_e, P_e, \tau, O_1, \dots, O_n, \pi_e \rangle$ where the components are as follows:

1. S_e is a (finite) set of *states of the environment*.
2. I_e is a subset of S_e , the *initial states* of the environment.
3. $P_e : S_e \rightarrow \mathcal{P}(ACT_e)$, the *protocol of the environment*,
4. τ is a function mapping joint actions $\mathbf{a} \in ACT_e \times ACT_1 \times \dots \times ACT_n$ to state transition functions $\tau(\mathbf{a}) : S_e \rightarrow S_e$.
5. For each $i = 1 \dots n$, the component O_i is a function, called the *observation function of agent i* , mapping the set of states S_e to some set \mathcal{O} .
6. $\pi_e : S_e \rightarrow \{0, 1\}^{Prop}$ is an *interpretation*,

Interpreted Systems

A *run* r of an environment E is an *infinite* sequence s_0, s_1, \dots of states such that $s_0 \in I_e$ and for all $m \geq 0$ there exists a joint action $\mathbf{a} = \langle a_e, a_1, \dots, a_n \rangle$ such that $s_{m+1} = \tau(\mathbf{a})(s_m)$ and $a_e \in P_e(s_m)$.

For $m \geq 0$ we write $r_e(m)$ for s_m .

A *point* is a tuple (r, m) , where r is a run and m a natural number.

An *interpreted system* for E is a tuple $\mathcal{I} = (\mathcal{R}, \pi)$ where

1. \mathcal{R} is a set of runs of E
2. $\pi : \text{Points}(\mathcal{R}) \rightarrow \{0, 1\}^{Prop}$ is defined by

$$\pi(r, m)(p) = \pi_e(r_e(m))(p)$$

Synchronous perfect-recall interpretation of knowledge

Given a run $r = s_0, s_1 \dots$ of an environment with observation functions O_i , we define the *local state of agent i at time $m \geq 0$* to be the sequence

$$r_i(m) = O_i(s_0) \dots O_i(s_m)$$

Define the equivalence relation \sim_i on points by

$$(r, m) \sim_i (r', m') \text{ if } r_i(m) = r'_i(m').$$

Semantics

Let \mathcal{I} be a system and (r, n) a point of \mathcal{I} .

1. $\mathcal{I}, (r, m) \models p$, where p is an atomic proposition, if $\pi(r, m)(p) = 1$,
2. $\mathcal{I}, (r, m) \models \varphi_1 \wedge \varphi_2$, if $\mathcal{I}, (r, m) \models \varphi_1$ and $\mathcal{I}, (r, m) \models \varphi_2$,
3. $\mathcal{I}, (r, m) \models \neg\varphi$, if not $\mathcal{I}, (r, m) \models \varphi$,
4. $\mathcal{I}, (r, m) \models \bigcirc\varphi$, if $\mathcal{I}, (r, m + 1) \models \varphi$,
5. $\mathcal{I}, (r, m) \models \varphi_1 U \varphi_2$, if there exists $k \geq m$ such that $\mathcal{I}, (r, k) \models \varphi_2$ and $\mathcal{I}, (r, l) \models \varphi_1$ for all l with $m \leq l < k$.

6. $\mathcal{I}, (r, m) \models K_i\varphi$, if $\mathcal{I}, (r', m') \models \varphi$ for all points (r', m') of \mathcal{I} satisfying $(r', m') \sim_i (r, m)$

System generated by a joint protocol

A *protocol* for agent i is a function $P_i : \mathcal{O}^+ \rightarrow ACT_i$.

A *joint protocol* \mathbf{P} is a tuple $\langle P_1, \dots, P_n \rangle$, where each P_i is a protocol for agent i .

The *interpreted system generated by a joint protocol \mathbf{P} in environment E* is the interpreted system $\mathcal{I}(\mathbf{P}, E) = (\mathcal{R}, \pi)$ for E where \mathcal{R} is the set of all runs of E generated by the agents executing joint protocol \mathbf{P} .

That is, \mathcal{R} is the set of runs r of E such that for all $m \geq 0$ we have $r_e(m+1) = \tau(\mathbf{a})(r_e(m))$, where

$$\mathbf{a} = \langle P_e(r_e(m)), P_1(r_1(m)), \dots, P_n(r_n(m)) \rangle$$

A joint protocol \mathbf{P} *realizes* a specification φ in an environment E if

$$\mathcal{I}(\mathbf{P}, E), (r, 0) \models \varphi$$

for all runs r of $\mathcal{I}(\mathbf{P}, E)$.

A specification φ is *realizable* in environment E if there exists a joint protocol \mathbf{P} that realizes φ in E .

Specification Examples

Auditability: $\Box(Bad \Rightarrow \Diamond K(Bad))$

Diagnosability: $\Box(K(Bad) \Rightarrow \Diamond K(Cause))$

Recoverability: $\Box(K(Bad) \Rightarrow \Diamond Good)$

Terminating Recoverability: $\Box(K(Bad) \Rightarrow \Diamond K(Good))$

Information Flow Security

Example: a multi-user operating system, shared by both High (top secret) and Low (public information) level users.

A *covert channel* is an unintended way for High information to flow to Low.

Example: High communicates to Low by locking/unlocking a shared peripheral

A spectrum of security properties

Can a Low user, passively observing the system, learn a High secret?

Can a Low user, acting on its own, behave in such a way that it can discover a High secret?

Can a High process (Trojan Horse) behave in such a way that a passive Low observer can discover a High secret?

Can a High process (Trojan Horse) behave in such a way that, in combination with a particular behaviour by Low, a High secret can leak to Low?

Passive Observation/Attack analysis

Let E be an environment for agents High and Low. Assume High and Low run fixed, finite state protocols P_L, P_H , respectively.

E.g. if Low is a passive observer and High is unconstrained, $P_L = \text{“do nothing”}$ and $P_H = \text{“do anything”}$.

Let p be a proposition the truth value of which is initially known only to High. Then the question

“Given that Low runs P_L and knows that High runs P_H , is Low guaranteed to eventually learn whether p ?”

amounts to

“Is $\diamond(K_{Low}p \vee K_{Low}\neg p)$ realized by $\langle P_L, P_H \rangle$ in E ?”

Model checking the logic of knowledge

Theorem [van der Meyden and Shilov, 97]: Given a formula φ of the logic of knowledge and linear time, a finite state environment E and a finite state joint protocol \mathbf{P} , it is decidable (with nonelementary complexity) whether φ is realized by \mathbf{P} in E (with respect to synchronous perfect recall).^a

Software: MCK: a model checker for the logic of knowledge and time

<http://www.cse.unsw.edu.au/~mck>

parameters:

- Linear/Branching time logic
- synchronous perfect recall, clock, observational semantics of knowledge

^aProof by reductions using Buchi automata, Weak SnS

Let E be an environment with two agents High, Low.

Let p be a proposition the truth value of which is initially known only to High.

Given a specific finite state protocol P_H for High, let E/P_H be the environment for the single agent Low obtained by fixing the actions of High according to P_H .

Then the question

“Given that Low knows High is running P_H , is there a behaviour of Low that guarantees that Low eventually learns whether p ?”

amounts to

“Is $\diamond(K_{Low}p \vee K_{Low}\neg p)$ realizable in E/P_H ?”

Theorem (van der Meyden and Vardi, 96): Let E be an environment for a single agent. There is an algorithm that decides whether a formula ψ in the logic of knowledge and (linear) time is realizable in E , in time^a

$$2^{O(\|E\|)} \cdot 2^{2^{O(\|\psi\|)}}.$$

The bound is tight, because

Theorem (Pnueli and Rosner 1989): Realizability of temporal logic formulae in systems with a single agent with complete information is 2-EXPTIME hard.

^aProof by reduction to alternating automata on infinite trees

Let E be an environment with two agents High, Low.

Let p be a proposition the truth value of which is initially known only to High.

Then the question

“Is there a Trojan Horse program, which if placed by Low at High, and acting in concert with some behaviour of Low, guarantees that Low eventually learns whether p ?”

amounts to

“Is $\diamond(K_{Low}p \vee K_{Low}\neg p)$ realizable in E ?”

Call this *weak non-deducibility on strategies* (cf. Wittbold & Johnson 90)

Theorem (Pnueli and Rosner 1989): Realizability of temporal logic formulae in systems with two agents with incomplete information is undecidable.

However, there exist classes of environments for multi-agent systems in which realizability of temporal logic is decidable.

Conjecture (van der Meyden and Vardi, 96): Realizability of formulae of the logic of knowledge and linear time is decidable in both hierarchical environments and broadcast environments .

Hierarchical Environments

An environment with n agents is *hierarchical* if for all states s, t , and agents $i = 1..n - 1$,

$$O_i(s) = O_i(t) \text{ implies } O_{i+1}(s) = O_{i+1}(t)$$

i.e., each agent i observes at least as much as agent $i + 1$.

Example:

agent 1 = High, observes top secret and public documents

agent 2 = Low, observes public documents only

Note: Pnueli and Rosner have a different notion of hierarchical architecture

Broadcast Environments

E is a *broadcast environment* for n agents if it has

- States $S = S_c \times S_1 \times \dots \times S_n$
- Common observations $O_c : S_c \rightarrow O$
- Observations $O_i((s_c, s_1, \dots, s_n)) = (O_c(s_c), s_i)$
- P_e depends on S_c only
- effect of joint action $(\mathbf{a}_e, \mathbf{a}_1, \dots, \mathbf{a}_n)$ on s_i depends on \mathbf{a}_i only

Example: a card game, with actions

- public announcements (bidding)
- playing a card

Recent results (van der Meyden and Wilke, CONCUR 2005)

Theorem: Realizability of formulae of the logic of knowledge and linear time in *broadcast environments* is *decidable* .

Theorem : Realizability of formulae of the logic of knowledge and linear time in *hierarchical environments* is *undecidable*.

Was the conjecture about hierarchical systems reasonable?

Theorem: Determining realizability of linear time logic formulas in hierarchical environments is decidable.

A reduction

A formula of the logic of knowledge and linear time is *positive* if it has only positive occurrences of formulas $K_i\varphi$.

Alternatively: the formula is constructed from basic formulas $p, \neg p$ using only $\wedge, \vee, K_i, \bigcirc, U, R$, (where $\alpha R\beta$ means $\neg(\neg\alpha U\neg\beta)$).

Theorem: There exists an efficiently computable transformation which given a positive formula φ and an environment E gives a formula φ' of linear time logic (without knowledge) and an environment E' such that φ is realizable in E iff φ' is realizable in E' . If E is hierarchical then so is E' .

Corollary: Realizability of positive formulas of the logic of knowledge and linear time in hierarchical environments is decidable.

Weak nondeducibility on strategies is decidable

Let E be a *hierarchical* environment with two agents High, Low.

Let p be a proposition the truth value of which is initially known only to High.

The question

“Is there a Trojan Horse program, which if placed by Low at High, and acting in concert with some behaviour of Low, guarantees that Low eventually learns whether p ?”

amounts to

“Is $\diamond(K_{Low}p \vee K_{Low}\neg p)$ realizable in E ?”

This is a positive formula, hence the question is decidable.

Can a Low user, passively observing the system, learn a high secret?
(**decidable, model checking, van der Meyden and Shilov, FSTTCS 97**)

Can a Low user, acting on its own, behave in such a way that it can discover a High secret? (**decidable, single agent synthesis**)

Can a High process (Trojan Horse) behave in such a way that a passive Low observer can discover a High secret? (**decidable, positive hierarchical synthesis**)

Can a High process (Trojan Horse) behave in such a way that, in combination with a particular behaviour by Low, a High secret can leak to Low?
(**decidable, positive hierarchical synthesis**)

Proof Sketches

Theorem: Realizability of formulae of the logic of knowledge and linear time in broadcast systems is decidable.

Idea: Reduce to an environment with one agent c , with observations the common observation. Apply single agent decidability.

Step 1: note that because of determinism, agents do not gain information from observation of private state

Step 2: eliminate initial private state by transformation of formula:

$$K_i(\varphi)^* = \bigvee_{s \in \text{Init}_i} p_{i,s} \wedge K_c(p_{i,s} \Rightarrow \varphi^*)$$

where the new proposition $p_{i,s}$ means "i's initial state was s ."

Comment: this would not work if protocols to be synthesized can be non-deterministic

Theorem: Realizability of formulae of the logic of knowledge and linear time in hierarchical systems is undecidable.

Proof by a reduction to a problem for Lossy Counter Machines.

A *lossy counter machine* is an automaton whose states are composed of

1. A control state C , taking a finite set of values
2. a set r_1, \dots, r_k of natural number valued registers.

The machine runs a finite set Δ of commands, of the types:

1. if $C = q$ then $r_j := r_j + 1, C := q'$
2. if $C = q$ and $r_j > 0$ then $r_j := r_j - 1; C := q_1$ else $C := q_2$

At each step of computation,

1. A subset of the registers r_j are decreased in value (non-deterministically)
2. the machine nondeterministically selects one of the instructions in Δ and executes it
3. A subset of the registers r_j are decreased in value (non-deterministically)

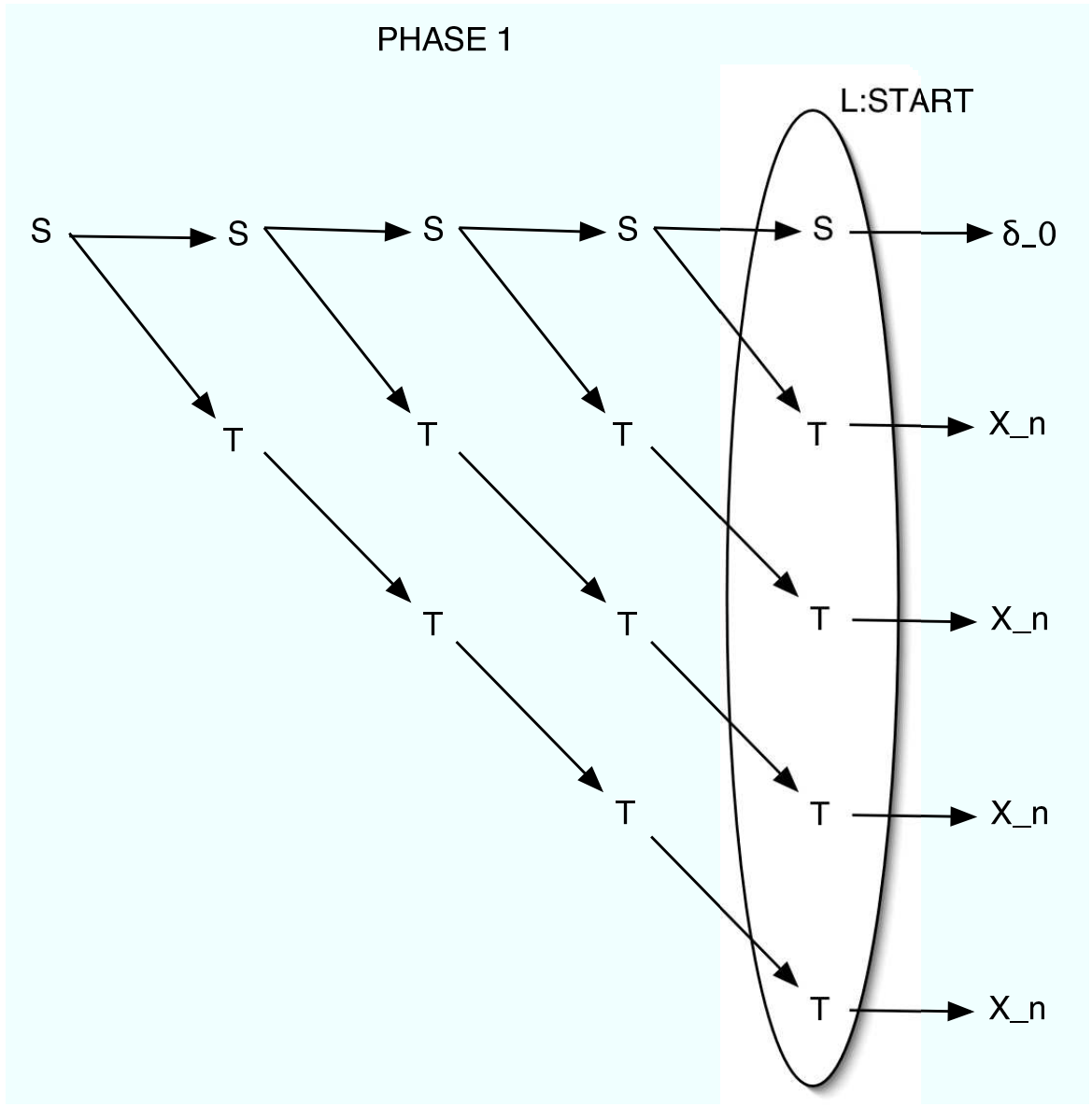
Theorem (Mayr 2003): The following problem is undecidable: Given a lossy counter machine M , is there a number n such that there exists an infinite run of M starting with the state $(C = q, r_1 = 0, r_2 = 0, \dots, r_k = n)$.

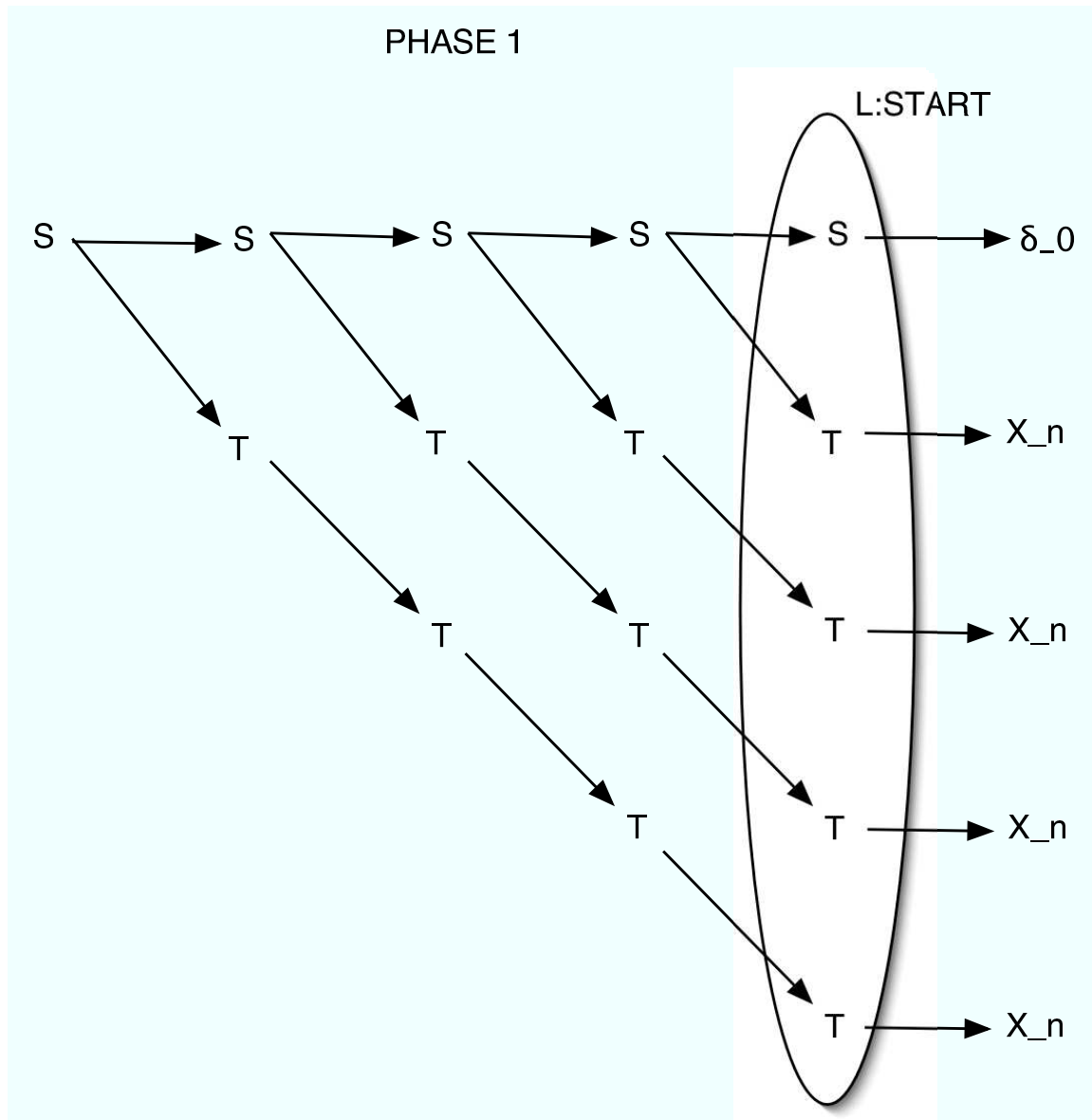
Idea of the reduction: Two agents:

High observes everything: $O_H(s) = s$

Low observes nothing: $O_L(s) = \perp$

Value of register r_j at time t is the number of possible sequences of observations (traces) of High ending in a state X_j





The formula checks whether the guesses are correct and the system $\mathcal{I}(E, \mathbf{P})$ corresponds to an infinite execution of the LCM:

if δ_k is “if $C = q$ and $r_j > 0$ then $r_j := r_j - 1; C := q_1$ else $C := q_2$ ” then δ_{k+1} must be of the form “if $C = q_1 \dots$ ” or “if $C = q_2 \dots$ ” (enforced by the environment, choice guessed by H)

To force H to make the correct guesses, the specification encodes

“if δ_k and $r_j > 0$ then
next(a decrement occurred on r_j and $C = q_1$) or next($C = q_2$)”

To check this, represent

1. $r_j > 0$ by $\neg K_L \neg X_j$
2. A decrement occurred on r_j by $\neg K_L \neg Dec_j$

Theorem 2: Determining realizability of linear time logic formulas in hierarchical environments is decidable.

An infinite (M, N) -tree is a tree with M -branching and N -node labels, that is, it is a function $M^* \rightarrow N$.

A protocol $P_i : O_i^* \rightarrow ACT_i$, which can be viewed as an (O_i, ACT_i) -tree

Base case:

Lemma: Given an environment E and a temporal specification φ , we can construct a finite tree automaton A for $(O_1, ACT_1 \times \dots \times ACT_n)$ -trees that accepts a tree t iff t realizes φ in the modified environment E' where agent 1 takes over the roles of agents 2 through n (and plays his own part).

Since the environment is hierarchical, for each pair of agents $i < j$, there exists a mapping $h_{i,j} : O_i \rightarrow O_j$ such that for all states s of the environment we have $O_j(s) = h_{i,j}(O_i(s))$.

Lemma: Given an environment E with just two agents and a tree automaton A for $(O_1, ACT_1 \times ACT_2)$ -trees, we can construct a finite tree automaton A' for (O_2, ACT_2) -trees such that A' accepts a tree t_2 iff there exists a protocol t_1 for agent 1 such that the joint protocol $(t_1, t_2 \circ h_{1,2})$ (viewed as a protocol for agent 1) is accepted by A .

Proof of the reduction

Theorem: There exists a computable transformation which given a positive formula φ of the logic of knowledge and linear time and an environment E gives a formula φ' of linear time logic (without knowledge) and an environment E' such that φ is realizable in E iff φ' is realizable in E' .

Idea: E' is like E , but in E' , agents also get to guess which knowledge subformulas of φ are true, and realizing φ' forces the protocol to guess correctly.

Let Φ_i be the set of all subformulas of φ of the form $K_i\alpha$.

Let ACT_i be the set of actions of i in E

Then in E' :

- The actions of agent i have the form (a, X_i) where $a \in ACT_i$ and $X_i \subseteq \Phi_i$. Intuitively, (a, X_i) represents “agent i does a and asserts that S is the set of i 's true knowledge subformulas of φ ”
- States are of the form (s, X_1, \dots, X_n) , consisting of a state s of E , plus a record of the last set of assertions X_i made by each agent i .
- $(s, X_1, \dots, X_n) \models P_{K_i\alpha}$ iff $K_i\alpha \in X_i$

Given ψ , define ψ^* to be the temporal logic formula obtained by replacing each maximal knowledge subformula $K_i\alpha$ by $\bigcirc P_{K_i}\alpha$.

E.g. $((K_1K_2p)UK_1q)^* = (\bigcirc P_{K_1K_2p})U(\bigcirc P_{K_1}q)$

Define

$$\varphi' = \varphi^* \wedge \bigwedge_{i=1..n, K_i\alpha \in \Phi_i} \square(\bigcirc P_{K_i}\alpha \Rightarrow \alpha^*)$$

Open problems/Ongoing work

1. Branching time versions
2. Synthesis of nondeterministic protocols
3. Complexity of specific security properties, tailored decision procedures
4. generalizations to probabilistic specifications/protocols