Team Semantics and Recursive Enumerability

Antti Kuusisto

University of Wrocław, Poland,
Technical University of Denmark
Stockholm University, Sweden
antti.j.kuusisto@uta.fi

Abstract. It is well known that dependence logic captures the complexity class NP, and it has recently been shown that inclusion logic captures P on ordered models. These results demonstrate that team semantics offers interesting new possibilities for descriptive complexity theory. In order to properly understand the connection between team semantics and descriptive complexity, we introduce an extension D^* of dependence logic that can define exactly all recursively enumerable classes of finite models. Thus D^* provides an approach to computation alterative to Turing machines. The essential novel feature in D^* is an operator that can extend the domain of the considered model by a finite number of fresh elements.

Keywords: team semantics, dependence logic, descriptive complexity

1 Introduction

In this article we study logics based on *team semantics*. Team semantics was originally conceived by Hodges [7] in the context of IF-*logic* [6]. On the intuitive level, team semantics provides an alternative compositional approach to systems based on game-theoretic semantics. The compositional approach simplifies the more traditional game-theoretic approaches in several ways.

In [13], Väänänen introduced *dependence logic* (D), which is a novel approach to IF-logic based on new atomic formulae $=(x_1, ..., x_k, y)$ that can be interpreted to mean that the choice for the value of y is *functionally determined* by the choices for the values of $x_1, ..., x_k$ in a semantic game.

After the introduction of dependence logic, research on logics based on team semantics has been *very active*. Several different logics with different applications have been investigated. Currently the two most important systems studied in the field in addition to dependence logic are *independence logic* [4] of Grädel and Väänänen and *inclusion logic* [2] of Galliani. Independence logic is a variant of dependence logic that extends first-order logic by new atomic formulae $x_1, ..., x_k \perp y_1, ..., y_n$ with the intuitive meaning that the interpretations of the variables $x_1, ..., x_k$ are *independent* of the interpretations of the variables $y_1, ..., y_n$. Inclusion logic extends first-order logic by atomic formulae $x_1, ..., x_k \subseteq y_1, ..., y_k$, whose intuitive meaning is that each tuple interpreting the

variables $x_1, ..., x_k$ must also be a tuple that interprets $y_1, ..., y_k$. Exclusion logic, also introduced in [2] by Galliani, is a natural counterpart of inclusion logic with atoms $x_1, ..., x_k \,|\, y_1, ..., y_k$ which state that the set of tuples interpreting $x_1, ..., x_k$ must not overlap with the set of tuples interpreting $y_1, ..., y_k$.

It was observed in [13] and [4] that dependence logic and independence logic are both equi-expressive with *existential second-order logic*, and thereby capture NP. Curiously, it was established in [3] that inclusion logic is equi-expressive with *greatest fixed point logic* and thereby captures P on finite ordered models. These results show that team semantics offers a novel interesting perspective on descriptive complexity theory. Especially the very close connection between team semantics and game-theoretic concepts is interesting in this context.

In order *properly understand* the perspective on descriptive complexity provided by team semantics, it makes sense to accomodate the related logics in a unified umbrella framework that exactly characterizes the computational capacity of Turing machines. It turns out that there exists a particularly simple extension of dependence logic that does the job. Let D^* denote the logic obtained by extending first-order logic by the atoms of dependence, independence, inclusion, and exclusion logic, and furthermore, an operator Ix that extends the domain of the model considered by a finite number of *fresh* elements. We show below that D^* can define exactly all recursively enumerable classes of finite models.

Since D^* captures RE, it is not only a logic but also a model of computation. The striking simplicity of D^* and the link between team semantics and game-theory make D^* a particularly interesting system. There of course exist other logical frameworks where RE can be easily captured, such as abstract state machines [5], [1] and the recursive games of [10]. However, D^* provides a *simple unified perspective on recent advances in descriptive complexity based on team semantics.* The framework of [10] resembles D^* since it provides a perspective on RE that explains computational notions via game-theoretic concepts, but the approach in [10] is burdened by potentially infinite games and [10] also lacks a compositional approach. The approach provided by D^* is at least in some reasonable sense more straighforward.

2 Preliminaries

We consider only models with a *purely relational vocabulary*, i.e., a vocabulary consisting of relation symbols only. Therefore, all vocabularies are below assumed to be purely relational without further warning. We let $\mathfrak{A}$, $\mathfrak{B}$, $\mathfrak{C}$, etc., denote models; A, B and C denote the domains of the models $\mathfrak{A}$, $\mathfrak{B}$ and $\mathfrak{C}$, respectively.

We let VAR denote a countably infinite set of exactly all first-order *variable symbols*. Let $X \subseteq \mathrm{VAR}$ be a *finite*, possibly empty set. Let A be a set. A function $s : X \to A$ is called an *assignment* with domain X and codomain A. We let $s[a/x]$ denote the assignment with domain $X \cup \{x\}$ and codomain $A \cup \{a\}$ defined such that $s[a/x](y) = a$ if $y = x$, and $s[a/x](y) = s(a)$ if $y \neq x$. Let T be a set. We define $s[\,T/x\,] \;=\; \{\; s[a/x] \mid a \in T \;\}$.

Let $X \subseteq \mathrm{VAR}$ be a finite, possibly empty set. Let U be a set of assignments $s : X \to A$. Such a set U is a *team* with *domain* X and codomain A. Note that the empty set is a team with codomain A, as is the set $\{\emptyset\}$ containing only the empty assignment. The team $\emptyset$ does not have a unique domain; any finite subset of VAR is a domain of $\emptyset$. The domain of the team $\{\emptyset\}$ is $\emptyset$. The domain of team U is denoted by $Dom(U)$.

Let T be a set. We define $U[\,T/x\,] := \{\ s[a/x] \mid a \in T,\ s \in U\ \}$. Let $f : U \to \mathcal{P}(T)$ be a function, where $\mathcal{P}$ denotes the power set operator. We define $U[\,f/x\,] := \bigcup_{s \in U} s[\,f(s)/x\,]$.

Let V be a team. Let $k \in \mathbb{Z}_+$, where $\mathbb{Z}_+$ denotes the positive integers. Let $x_1, ..., x_k \in Dom(V)$. Define $Rel\big(V, (x_1, ..., x_k)\big) := \{\big(s(x_1), ..., s(x_k)\big) \mid s \in V\}$.

We then define *lax team semantics* for formulae of first-order logic (FO). As usual in investigations related to team semantics, formulae are assumed to be in *negation normal form*, i.e., negations occur only in front of atomic formulae. Let $\mathfrak{A}$ be a model and U a team with codomain A. Let $\models_{\mathrm{FO}}$ denote the ordinary Tarskian satisfaction relation of first-order logic, i.e., $\mathfrak{A}, s \models_{\mathrm{FO}} \varphi$ means that the model $\mathfrak{A}$ satisfies the first-order formula φ under the assignment s. We define

$$
\begin{array}{lll}
\mathfrak{A}, U \models x = y & \Leftrightarrow & \forall s \in U\big(\mathfrak{A}, s \models_{\mathrm{FO}} x = y\big), \\
\mathfrak{A}, U \models \neg x = y & \Leftrightarrow & \forall s \in U\big(\mathfrak{A}, s \models_{\mathrm{FO}} \neg x = y\big), \\
\mathfrak{A}, U \models R(x_1, ..., x_k) & \Leftrightarrow & \forall s \in U\big(\mathfrak{A}, s \models_{\mathrm{FO}} R(x_1, ..., x_k)\big), \\
\mathfrak{A}, U \models \neg R(x_1, ..., x_k) & \Leftrightarrow & \forall s \in U\big(\mathfrak{A}, s \models_{\mathrm{FO}} \neg R(x_1, ..., x_k)\big), \\
\mathfrak{A}, U \models (\varphi \wedge \psi) & \Leftrightarrow & \mathfrak{A}, U \models \varphi \text{ and } \mathfrak{A}, U \models \psi, \\
\mathfrak{A}, U \models (\varphi \vee \psi) & \Leftrightarrow & \mathfrak{A}, U_0 \models \varphi \text{ and } \mathfrak{A}, U_1 \models \psi \text{ for some} \\
 & & \text{teams } U_0, U_1 \subseteq U \text{ such that } U_0 \cup U_1 = U, \\
\mathfrak{A}, U \models \forall x\, \varphi & \Leftrightarrow & \mathfrak{A}, U[\,A/x\,] \models \varphi, \\
\mathfrak{A}, U \models \exists x\, \varphi & \Leftrightarrow & \mathfrak{A}, [\,f/x\,] \models \varphi \text{ for some } f : U \to (\mathcal{P}(A) \setminus \emptyset).
\end{array}
$$

A sentence φ is true in $\mathfrak{A}$ ($\mathfrak{A} \models \varphi$) if $\mathfrak{A}, \{\emptyset\} \models \varphi$. It is well known and easy to show that for an FO-formula φ, we have $\mathfrak{A}, U \models \varphi$ iff $\mathfrak{A}, s \models_{\mathrm{FO}} \varphi$ for all $s \in U$.

Proposition 1. *Let φ be a formula of first-order logic. Let U be a team. Then $\mathfrak{A}, U \models \varphi$ iff $\forall s \in U(\mathfrak{A}, s \models_{\mathrm{FO}} \varphi)$.* □

Dependence logic (D) is the extension of first-order logic in negation normal form with novel atoms $=(x_1, ..., x_k)$ for each positive integer k. These atoms are called *dependence atoms*. The semantics dictates that $\mathfrak{A}, U \models =(x_1, ..., x_k)$ iff for each $s, t \in U$ such that $s(x_i) = t(x_i)$ for each $i \in \{1, ..., k-1\}$, we have $s(x_k) = t(x_k)$. We note that dependence logic is sometimes formulated such that negated atoms $\neg=(x_1, ..., x_k)$ are allowed, but since the semantics then dictates that $\mathfrak{A}, U \models \neg=(x_1, ..., x_k)$ iff $U = \emptyset$, these negated atoms can be replaced by $\exists x(x \neq x)$.

Inclusion logic is obtained by extending first-order logic in negation normal form by atoms $x_1, ..., x_k \subseteq y_1, ..., y_k$ with the semantics $\mathfrak{A}, U \models\ x_1, ..., x_k \subseteq y_1, ..., y_k$ iff $Rel(U, (x_1, ..., x_k)) \subseteq Rel(U, (y_1, ..., y_k))$. Here k can be any positive integer. Similarly, *exclusion logic* extends first-order logic in negation normal

form with atoms $x_1, ..., x_k \,|\, y_1, ..., y_k$ such that $\mathfrak{A}, U \models\ \ x_1, ..., x_k \,|\, y_1, ..., y_k$ iff $Rel(U, (x_1, ..., x_k)) \cap Rel(U, (y_1, .., y_k)) = \emptyset$. Again k can be any positive integer. *Independence logic* extends first-order logic in negation normal form with atoms $x_1, ..., x_k \perp_{z_1,...,z_m} y_1, ..., y_n$ such that $\mathfrak{A}, U \models x_1, ..., x_k \perp_{z_1,...,z_m} y_1, ..., y_n$ iff for all $s, s' \in U$ there exists a $t \in U$ such that

$$\big(\bigwedge_{i \leq m} s(z_i) = s'(z_i) \big) \Rightarrow \big(\bigwedge_{i \leq k} t(x_i) = s(x_i) \wedge \bigwedge_{i \leq m} t(z_i) = s(z_i) \wedge \bigwedge_{i \leq n} t(y_i) = s'(y_i) \big).$$

Here k, m, n can be any positive integers. Independence logic also contains atoms $x_1, ..., x_k \perp y_1, ..., y_n$ such that $\mathfrak{A}, U \models x_1, ..., x_k \perp y_1, ..., y_n$ iff for all $s, s' \in U$ there exists a $t \in U$ such that $\bigwedge_{i \leq k} t(x_i) = s(x_i)$ and $\bigwedge_{i \leq n} t(y_i) = s'(y_i)$. Here k and n can be any positive integers.

Let $\mathfrak{A}$ be a model and τ its vocabulary. Let $S \neq \emptyset$ be finite a set such that $S \cap A = \emptyset$. We let $\mathfrak{A} + S$ denote the model $\mathfrak{B}$ such that $B = A \cup S$ and $R^{\mathfrak{B}} = R^{\mathfrak{A}}$ for all $R \in \tau$. The model $\mathfrak{B}$ is called a *finite bloating* of $\mathfrak{A}$.

We then define the logic D^* that captures recursive enumerability. In the spirit of team semantics, D^* is based on the use of sets of assignments, i.e., teams, that involve first-order variables. Let D^+ denote the logic obtained by extending first-order logic in negation normal form by all dependence atoms, independence atoms, inclusion atoms, and exclusion atoms. D^* is obtained by extending D^+ by an additional formula formation rule stating that if φ is a formula, then so is $\mathrm{I}x\,\varphi$. We define $\mathfrak{A}, U \models \mathrm{I}x\,\varphi$ iff there exists a finite bloating $\mathfrak{A} + S$ of $\mathfrak{A}$ such that $\mathfrak{A} + S, U[S/x] \models \varphi$. We note that there are connections between different classes of atoms: for example, since $=(x_1, ..., x_k, y)$ is equivalent to $y \perp_{x_1,...,x_k} y$, dependence atoms can in fact be very easily eliminated from D^*.

Note that if desired, we can avoid reference to a proper class of possible bloatings of $\mathfrak{A}$ in the semantics of D^* by letting $A_1 := A \cup \{A\}$ to be the canonical bloating of A by one element and $A_{k+1} := A_k \cup \{A_k\}$ the bloating of A by $k + 1$ elements.

3 D* Captures RE

Let τ be a vocabulary. Sentences of *existential second-order logic* (ESO) over τ are formulae of the type $\exists X_1 ... \exists X_k\, \varphi$, where $X_1, ..., X_k$ are relation variables and φ a sentence of FO over $\tau \cup \{X_1, ..., X_k\}$. The symbols $X_1, ..., X_k$ are not in τ. We extend ESO by defining a logic $\mathcal{L}_{RE}$, whose τ-sentences are of the type $\mathrm{I}Y\,\psi$, where $Y \notin \tau$ is a *unary* relation variable and ψ an ESO-sentence over $\tau \cup \{Y\}$. Let $\mathfrak{A}$ be a τ-model. The semantics of $\mathcal{L}_{RE}$ is defined such that $\mathfrak{A} \models \mathrm{I}Y\,\psi$ iff there exists a finite set $S \neq \emptyset$ such that the following conditions hold.

1. $A \cap S = \emptyset$.
2. Let $\mathfrak{A}^+$ be the model of the vocabulary $\tau \cup \{Y\}$ with domain $A \cup S$ such that $Y^{\mathfrak{A}^+} = S$ and $R^{\mathfrak{A}^+} = R^{\mathfrak{A}}$ for all $R \in \tau$. We have $\mathfrak{A}^+ \models \psi$.

As we shall see, the logic $\mathcal{L}_{RE}$ can define in the finite exactly all recursively enumerable classes of finite models.

Let $\sigma \neq \emptyset$ be a finite set of unary relation symbols and *Succ* a binary relation symbol. A *word model* over the vocabulary $\{Succ\} \cup \sigma$ is a model $\mathfrak{A}$ defined as follows.

1. The domain A of $\mathfrak{A}$ is a nonempty finite set. The predicate *Succ* is a successor relation over A, i.e., a binary relation corresponding to a linear order, but with maximum out-degree and in-degree equal to one.
2. Let $b \in A$ be the smallest element with respect to *Succ*. We have $b \notin P^{\mathfrak{A}}$ for all $P \in \sigma$. (This is because we do not allow models with the empty domain; the empty word corresponds to the word model with exactly one element.) For all $a \in A \setminus \{b\}$, there is exactly one $P \in \sigma$ such that $a \in P^{\mathfrak{A}}$.

Word models canonically encode finite words. For example the word *abbaa* over the alphabet $\{a, b\}$ is encoded by the word model $\mathfrak{M}$ over the vocabulary $\{Succ, P_a, P_b\}$ defined such that $M = \{0, ..., 5\}$ and $Succ^{\mathfrak{M}}$ is the canonical successor relation on M, and we have $P_a^{\mathfrak{M}} = \{1, 4, 5\}$ and $P_b^{\mathfrak{M}} = \{2, 3\}$.

When investigating computations on structure classes (rather than strings), Turing machines of course operate on *encodings* of structures. We will use the encoding scheme of [11]. Let τ be a finite vocabulary and $\mathfrak{A}$ a finite τ-structure. In order to encode the structure $\mathfrak{A}$ by a binary string, we first need to define a linear ordering of the domain A of $\mathfrak{A}$. Let $<^{\mathfrak{A}}$ denote such an ordering.

Let $R \in \tau$ be a k-ary relation symbol. The encoding $enc(R^{\mathfrak{A}})$ of $R^{\mathfrak{A}}$ is the $|A|^k$-bit string defined as follows. Consider an enumeration of all k-tuples over A in the *lexicographic order* defined with respect to $<^{\mathfrak{A}}$. In the lexicographic order, $(a_1, ..., a_k)$ is smaller than $(a'_1, ..., a'_k)$ iff there exists $i \in \{1, ..., k\}$ such that $a_i < a'_i$ and $a_j = a'_j$ for all $j < i$. There are $|A|^k$ tuples in A^k, and the string $enc(R^{\mathfrak{A}})$ is the word $t \in \{0, 1\}^*$ of the length $|A|^k$ such that the bit t_i of $t = t_1 ... t_{|A|^k}$ is 1 if and only if the i-th tuple $(a_1, ..., a_k) \in A^k$ in the lexicographic order is in the relation $R^{\mathfrak{A}}$.

The encoding $enc(\mathfrak{A})$ is defined as follows. We first order the relations in τ. Let p be the number of relations in τ, and let $R_1, ..., R_p$ enumerate the symbols in τ according to the order. We define $enc(\mathfrak{A}) := 0^{|A|} \cdot 1 \cdot enc(R_1^{\mathfrak{A}}) \cdot ... \cdot enc(R_p^{\mathfrak{A}})$. Notice that the encoding of $\mathfrak{A}$ indeed depends on the order $<^{\mathfrak{A}}$ and the ordering of the relation symbols in τ, so $\mathfrak{A}$ in general has several encodings. However, we assume that τ is always ordered in some canonical way, so the multiplicity of encodings results in only due to different orderings of the domain of $\mathfrak{A}$.

Let τ be a finite vocabulary. A Turing machine TM defines a *semi-decision algorithm for a class $\mathcal{C}$ of finite τ-models* iff there is an accepting run for TM on an input $w \in \{0, 1\}^*$ exactly when w is some encoding of some structure $\mathfrak{A} \in \mathcal{C}$.

Proposition 2. *In the finite, $\mathcal{L}_{RE}$ can define exactly all recursively enumerable classes of models.*

Proof. Let TM be a Turing machine that defines a semi-decision algorithm for some class of models. It is routine to write a formula $\varphi_{\mathrm{TM}} := \mathrm{I}Y\overline{\exists X}\, \psi$ such that $\mathfrak{A} \models \varphi_{\mathrm{TM}}$ if there exists an extension $\mathfrak{B}$ of $\mathfrak{A}$ that consist essentially of

a copy of $\mathfrak{A}$ and another part $\mathfrak{C}$ that encodes the computation table of an accepting computation of TM on an input $enc(\mathfrak{A})$. We can use the predicates in $\overline{\exists X}$ in order to define word models that encode $enc(\mathfrak{A})$ and other strings that correspond to the Turing machine tape at different stages of the computation. Symbols in $\overline{\exists X}$ can also be used, inter alia, in order to define the other parts of the computation table and an ordering of the domain of $\mathfrak{A}$, and also relations that connect $\mathfrak{A}$ to $\mathfrak{C}$ in order to ensure $\mathfrak{A}$ and $\mathfrak{C}$ are correctly related. The symbol Y is used in order to see which points belong to the original model $\mathfrak{A}$.

For the converse, given a sentence $\mathrm{I}Y\overline{\exists X}\,\psi$ of $\mathcal{L}_{RE}$, we can define a Turing machine that first non-deterministically provides a number $k \in \mathbb{Z}_+$ of fresh points to be added to the domain of the model considered, and then checks if $\overline{\exists X}\,\psi$ holds in the obtained larger model. □

Our next aim is to discuss Lemma 1, which essentially provides a way of encoding a unary relation symbol Y by a corresponding variable symbol y with the help of inclusion, exclusion, and independence atoms. For the purposes of the Lemma, we first define a translation from dependence logic to D^+. Rönnholm considers a translation with similar intuitions in [12].

Let χ be a *sentence* of dependence logic over a vocabulary τ such that $Y \notin \tau$. Let y, v, u, u' be variables that *do not occur* in χ. We next define a translation $T_Y^y(\chi)$ of χ into D^+ by recursion on the structure of χ. (Strictly speaking, the variables v, u, u' are fixed parameters of the translation just like y and Y, so we should write $T_Y^{y,v,u,u'}(\chi)$ instead of $T_Y^y(\chi)$. The issue here is only that when a sentence χ is translated, the auxiliary variables y, v, u, u' should not occur in χ.)

1. $T_Y^y(R(x_1,...,x_k)) := R(x_1,...,x_k)$ and $T_Y^y(\neg R(x_1,...,x_k)) := \neg R(x_1,...,x_k)$
2. $T_Y^y(x = z) := x = z$ and $T_Y^y(\neg x = z) := \neg x = z$
3. $T_Y^y(Y(x)) := x \subseteq y$ and $T_Y^y(\neg Y(x)) := x|y$
4. $T_Y^y(=(x_1,...,x_k)) := =(x_1,...,x_k)$
5. $T_Y^y((\varphi \wedge \psi)) := ((T_Y^y(\varphi) \wedge T_Y^y(\psi))$
6. $T_Y^y((\varphi \vee \psi)) := \exists v\big(v \bot_{\overline{z}} y \wedge \big((T_Y^y(\varphi) \wedge v = u) \vee (T_Y^y(\psi) \wedge v = u')\big)\big)$, where $\overline{z}$ contains exactly all variables quantified superordinate to $(\varphi \vee \psi)$ in χ, i.e., exactly each x such that $(\varphi \vee \psi)$ is in the scope of $\exists x$ or $\forall x$.
7. Assume $\exists x\,\varphi$ is subordinate to a disjunction in χ, meaning that there is a subformula $(\alpha \vee \beta)$ of χ and $\exists x\,\varphi$ is a subformula of $(\alpha \vee \beta)$. We define $T_Y^y(\exists x\,\varphi) := \exists x\big(x \bot_{\overline{z}} yv \wedge T_Y^y(\varphi)\big)$, where $\overline{z}$ contains exactly all variables quantified superordinate to $\exists x\,\varphi$ in χ, with the exception that $\overline{z}$ never contains x; the exception is relevant if χ contains nested quantification of x.
8. Assume $\exists x\,\varphi$ is not subordinate to a disjunction in χ. Then $T_Y^y(\exists x\,\varphi) := \exists x\big(x \bot_{\overline{z}} y \wedge T_Y^y(\varphi)\big)$, where $\overline{z}$ contains exactly all variables quantified superordinate to $\exists x\,\varphi$ in χ, with the exception that $\overline{z}$ never contains x.
9. $T_Y^y(\forall x\,\varphi) := \forall x\,(T_Y^y(\varphi))$

Let $\mathfrak{A}$ be a model such that $|A| \geq 2$. Let $S \subseteq A$. Let χ be a sentence of dependence logic and φ a subformula of χ. Let (U, V) a pair of be teams with codomain A such that the following conditions hold.

1. Call $Z := Dom(U)$. Z contains exactly all variables quantified superordinate to φ in χ. $Dom(V)$ is $Z \cup \{y, u, u'\}$ or $Z \cup \{y, v, u, u'\}$; we have $v \in Dom(V)$ iff φ is subordinate to a disjunction in χ.
2. We have $U = V \upharpoonright Z$, i.e., $U = \{\, s \upharpoonright Z \mid s \in V \,\}$, where $Z = Dom(U)$.
3. There exists a team X such that $V = X[S/y]$. (Thus $S \neq \emptyset$ if $V \neq \emptyset$.)
4. For all $s, t \in V$, we have $s(u) = t(u) \neq t(u') = s(u')$. In other words, every assignment in V gives exactly the same interpretation to u and to u', and the interpretation of u is different from that of u'.

When (U, V) satisfies the above four conditions, we say that (U, V) is a *suitable pair* for $\mathfrak{A}$, $S \subseteq A$, (y, v, u, u') and (φ, χ). Below $\mathfrak{A}$, S, and (y, v, u, u') will always be clear from the context (and in fact the same everywhere), so we may simply talk about suitable pairs for (φ, χ).

Let $\mathfrak{B}$ be a model and $T \subseteq B$ a set. Let τ be the vocabulary of $\mathfrak{B}$. Let $P \notin \tau$ be a unary relation symbol. We let $(\mathfrak{B}, P \mapsto T)$ denote the expansion of $\mathfrak{B}$ to the vocabulary $\tau \cup \{P\}$ such that $P^{\mathfrak{B}} = T$.

Let s be an assignment with domain X. Let $\{x_1, ..., x_k\}$ be a finite set of variables. We let $s_{-\{x_1,...,x_k\}}$ denote the assignment $s \upharpoonright (X \setminus \{x_1, ..., x_k\})$.

Lemma 1. *Let χ be a sentence of dependence logic not containing the symbols y, v, u, u'. Let $\mathfrak{A}$ be a model with at least two elements. Let Y be a unary relation symbol that occurs neither in χ nor in the vocabulary of $\mathfrak{A}$. Let $S \subseteq A$. Let $(\{\emptyset\}, V)$ be a suitable pair of for $\mathfrak{A}$, S, (y, v, u, u') and (χ, χ). Then we have $\big(\mathfrak{A}, Y \mapsto S\big), \{\emptyset\} \models \chi$ iff $\mathfrak{A}, V \models T^y_Y(\chi)$.*

Proof. Prove by induction on the structure of χ that for any subformula φ of χ, the equivalence $\big(\mathfrak{A}, Y \mapsto S\big), U \models \varphi \Leftrightarrow \mathfrak{A}, V \models T^y_Y(\varphi)$ holds for all suitable pairs (U, V) for $\mathfrak{A}$, S, (y, v, u, u') and (φ, χ). □

Define $\mathbb{T}^y_Y(\varphi) := \exists u \exists u' \big(u \neq u' \wedge =\!(u) \wedge =\!(u') \wedge T^y_Y(\varphi) \big)$. The following Lemma now follows directly.

Lemma 2. *Let $\mathfrak{A}$ be a model such that $|A| \geq 2$. Let $S \subseteq A$ be a nonempty finite set. Let φ be a sentence of dependence logic. Let y be a variable that does not occur in φ. Let Y be a unary symbol that occurs neither in φ nor in the vocabulary of $\mathfrak{A}$. Then $(\mathfrak{A}, Y \mapsto S), \{\emptyset\} \models \varphi$ iff $\mathfrak{A}, \{\emptyset\}[S/y] \models \mathbb{T}^y_Y(\varphi)$.* □

Theorem 1. *$\mathcal{L}_{RE}$ is contained in* D^*.

Proof. It is well known that every sentence α of ESO translates to an equivalent sentence $\alpha^{\#}$ of dependence logic, see [13]. We shall use this translation below.

Let $\varphi := \mathrm{I} Y \, \overline{\exists X} \psi$ be a sentence of $\mathcal{L}_{RE}$, where ψ is a first-order sentence. The following chain of equivalences, where the penultimate equivalence follows by Lemma 2, settles the current theorem.

$$
\begin{aligned}
\mathfrak{A} \models \varphi &\Leftrightarrow \big(\mathfrak{A} + S, Y \mapsto S\big) \models \overline{\exists X} \psi \text{ for some finite } S \neq \emptyset \text{ s.t. } S \cap A = \emptyset \\
&\Leftrightarrow \big(\mathfrak{A} + S, Y \mapsto S\big), \{\emptyset\} \models \big(\overline{\exists X} \psi\big)^{\#} \text{ for some finite } S \neq \emptyset \text{ s.t. } S \cap A = \emptyset \\
&\Leftrightarrow \mathfrak{A} + S, \{\emptyset\}[S/y] \models \mathbb{T}^y_Y\big(\big(\overline{\exists X} \psi\big)^{\#}\big) \text{ for some finite } S \neq \emptyset \text{ s.t. } S \cap A = \emptyset \\
&\Leftrightarrow \mathfrak{A}, \{\emptyset\} \;\models\; \mathrm{I} y \, \mathbb{T}^y_Y\big(\big(\overline{\exists X} \psi\big)^{\#}\big)
\end{aligned}
$$

□

Theorem 2. D^* *is contained in* $\mathcal{L}_{RE}$.

Proof. Let φ be a sentence of D^*. Assume φ contains k occurrences of the operator I. Let TM be a Turing machine such that when given an input model $\mathfrak{A}$, TM first nondeterministically constructs a tuple $\overline{n} \in (\mathbb{Z}_+)^k$ that gives for each occurrence of I in φ a number of new points to be added to the model. Then TM checks whether $\mathfrak{A}$ satisfies φ with the given tuple $\overline{n}$ of cardinalites to be added during the evaluation. TM is a semi-decision algorithm corresponding to φ. □

4 Conclusions

We have shown how the standard logics based on team semantics extend naturally to the simple system D^* that captures RE. The system D^* nicely expands the scope of team semantics from logic to computation. It will be interesting to investigate, for example, *what kind of decidable fragments* D^* *has.* Furthermore, it would be interesting to investigate generalized quantifiers and generalized atoms ([8,9]) in the context of D^*.

References

1. Börger, E., Stärk, R.F.: Abstract State Machines. A Method for High- Level System Design and Analysis. Springer (2003)
2. Galliani, P.: Inclusion and exclusion dependencies in team semantics - on some logics of imperfect information. Ann. Pure Appl. Logic, 163(1), pp. 68-84, (2012)
3. Galliani, P., Hella, L.: Inclusion logic and Fixed point logic. In: CSL 2013, pp. 281-295 (2013)
4. Grädel, E., Väänänen, J.: Dependence and independence. Studia Logica, 101(2), pp. 399-410 (2013)
5. Gurevich, Y.: A new thesis. American Mathematical Soc. Abstracts, 6(4), p. 317 (1985)
6. Hintikka, J., Sandu, G.: Informational independence as a semantical phenomenon. In: Logic, Methodology and Philosophy of Science VIII, Elsevier (1989)
7. Hodges, W.: Compositional semantics for a language of imperfect information. Logic Journal of the IGPL, 5, 539-563 (1997)
8. Kuusisto, A.: Logics of incomplete information without identity. TamPub, University of Tampere (2011)
9. Kuusisto, A.: A double team semantics for generalized quantifiers. CoRR, abs/1310, 3032 (2013)
10. Kuusisto, A.: Some Turing-complete extensions of First-order logic. CoRR, abs/1405,1715 (2014)
11. Libkin, L.: Elements of Finite Model Theory. Springer (2004)
12. Rönnholm, R.: Inkluusio ja ekskluusio kvantifioinnissa. TamPub, University of Tampere (2014)
13. Väänänen, J.: Dependence logic: A new approach to independence friendly logic. Cambridge University Press (2007)