Team Logic[*]

Jouko Väänänen

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam, The Netherlands

Matematiikan ja tilastotieteen laitos
Helsingin yliopisto
P.O. Box 68
00014 Helsinki, Finland

vaananen@science.uva.nl

Abstract

Team logic is the logic of functional dependencies. We define the basic logical operations of team logic, establish its relationship with independence friendly and second order logic, and give it a game theoretic characterization.

1 Introduction

Let a vocabulary[1] L and an L-structure $\mathcal{M}$ with universe M be given. In this paper we study *functional dependencies* in $\mathcal{M}$. This is in contrast to the traditional approach in logic of studying *relational dependencies* in $\mathcal{M}$. Our atomic dependence relations state the existence of a functional dependence without giving any definition for the function that carries the dependence. This gives the whole topic a second order flavor. It seems to the author that although functional dependence in databases has been studied (starting with [Ar74]), a general theory of more complex types of dependence is new. (For a more detailed recent study, see [Vä07].) Our theory is based on [Ho$_1$97b] and [VäHo$_1\infty$].

If L has an n-ary function symbol f, there is immediately an apparent dependence relation in $\mathcal{M}$, namely the dependence of each $a = f^{\mathcal{M}}(a_1, \dots, a_n)$ on $a_1, \dots, a_n$. If the function $f^{\mathcal{M}}$ is constant, the dependence is of a singular kind, not commonly called dependence on $a_1, \dots, a_n$ at all. On the

* This paper was written while the author was a guest in the Newton Institute (Cambridge, U.K.) as part of the special program *Logic and Computation.* The author is grateful to the Newton Institute for its support. Research was also partially supported by grant 40734 of the Academy of Finland. I am indebted to the referee for pointing out some inaccuracies and suggesting improvements, and to Fred Oakley for permission to use the photo in Figure 2.

[1] The vocabulary may contain constant, relation and function symbols.

other hand, if the function is one to one, we have a dependence that is so perfect that it can even be reversed. In this case dependence is carried by a function that has a name in the vocabulary. In general we have completely abstract functions carrying the dependence with no name in the vocabulary and even no definition in the language, what so ever.

If L has an n-ary relation symbol R, there is a relational dependence relation in $\mathcal{M}$, namely the mutual dependence of $a_1, \ldots, a_n$ such that $(a_1, \ldots, a_n) \in R^{\mathcal{M}}$ on each other. However, even if such a relation between the elements $a_1, \ldots, a_n$ binds the elements together in an obvious sense, it need not in general be a (functional) dependence in the sense of the current approach.

Team logic is the logic of (functional) dependence. We define the basic concepts of team logic and use it to analyze dependence. The negation-free part of team logic turns out to correspond naturally to independence friendly logic and thereby to the existential part of second order logic. Team logic itself can be seen as being the natural closure of independence friendly logic under (classical) negation. In expressive power team logic is in a natural sense equivalent to full second order logic. We give a game-theoretic characterization of team logic by means of an appropriate Ehrenfeucht-Fraïssé game.

2 Agents and teams

An *agent* is any finite mapping s from a domain $\mathrm{dom}(s)$ into M. Elements of $\mathrm{dom}(s)$ are called *features* or *fields*, sometimes *attributes*, depending on the application[2]. Unless stated otherwise, to be specific, the domain of an agent is a finite set of natural numbers. The *modification* $s(a/n)$ of an agent s maps n to a but agrees otherwise with s. The *value* of a term t, built up from variables x_n and symbols of L, on an agent s is the element $t\langle s\rangle$ of M defined in the usual way by $c\langle s\rangle = c^{\mathcal{M}}$, $x_n\langle s\rangle = s(n)$, and $ft_1 \ldots t_n\langle s\rangle = f^{\mathcal{M}}(t_1\langle s\rangle, \ldots, t_n\langle s\rangle)$. For this to make sense the domain of s has to contain n whenever x_n occurs in t.

Building on [Ho₁97a] and [Ho₁97b] we take the position that dependence, just like any other pattern, cannot be manifested by one event, observation or agent (which is our terminology) but needs a series or a group of events, observations or agents. We call these series or groups manifesting dependence teams.

A *team* (see Figure 1) is any set X of agents with the same domain $\mathrm{dom}(X)$. Table 1 presents a generic team in the form of a table. Here are some examples of teams:

[2] Agents are also called assignments or tuples. If we think of an agent s as an assignment, the elements of the domain are variables x_n or their indexes n, depending on how values of terms are defined.

	Features			
Agent	m_1	m_2	$\ldots$	m_n
s_1	$s_1(m_1)$	$s_1(m_2)$	$\ldots$	$s_1(m_n)$
s_2	$s_2(m_1)$	$s_2(m_2)$	$\ldots$	$s_2(m_n)$
s_3	$s_3(m_1)$	$s_3(m_2)$	$\ldots$	$s_3(m_n)$
s_4	$s_4(m_1)$	$s_4(m_2)$	$\ldots$	$s_4(m_n)$
s_5	$s_5(m_1)$	$s_5(m_2)$	$\ldots$	$s_5(m_n)$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$

TABLE 1. A generic team as a table.

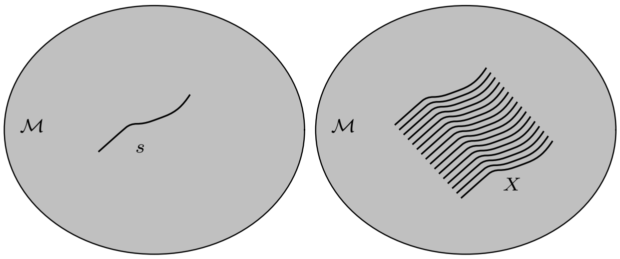

FIGURE 1. A model with an assignment and a model with a team.

Team 1 A team of human genomes (Figure 2). Here the agents are individual genomes of individual people. The fields (or features) are the individual genes that are observed. Potentially there are tens of thousands of possible fields to consider, and billions of agents. If we add to such a team new fields related to medical data of the person in question, we get a team the dependencies of which may give crucial data about the role of hereditary factors in medical science. For example, we may ask:

1. Does a certain gene or gene combination (significantly) *determine* a given hereditary disease in the sense that a patient with (a fault in) those genes has a high risk of the disease?
2. Is a disease *totally dependent* on a gene in the sense that every gene combination that (significantly) determines the disease contains that particular gene.
3. Is a gene (merely) *dependent* on a gene in the sense that the

disease is (significantly) determined by some gene combination with the gene but not without.

4. Is a disease *totally independent* of a gene in the sense that no gene combination that (significantly) determines the disease contains that particular gene.
5. Is a gene (merely) *independent* of a gene in the sense that some gene combinations (significantly) determine the disease without containing that particular gene.

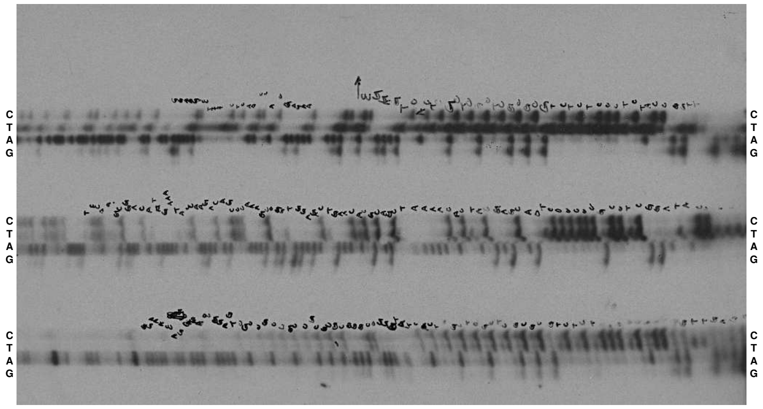

FIGURE 2. A piece of a team of genomes.

Team 2 Game history team: Imagine a game, any game. Game moves are the features of this team, plays are the agents, and a collection of plays is a team. Thus a player is identified with his or her behavior in the game. It may be relevant to know answers to the following kinds of questions:

1. What is the strategy that a player (e.g. "Nature") is following, or is he or she following any strategy at all?
2. Is a player using information about his or her (or other players') moves that he or she is committed not to use?

Team 3 Every formula $\varphi(x_1, \ldots, x_n)$ of any logic and structure $\mathcal{M}$ give rise to the team of all assignments that satisfy $\varphi(x_1, \ldots, x_n)$ in $\mathcal{M}$. This is a definable team and perhaps the most obvious team arising in logic. It may manifest functional dependencies on the structure, and depending on the logic, these dependencies can or cannot be expresses in the logic.

Team 4 Every first order sentence φ and structure $\mathcal{M}$ give rise to teams consisting of assignments that arise in the semantic game of φ and $\mathcal{M}$. If $\mathcal{M} \models \varphi$ and the winning strategy of II is τ, a particularly coherent team consists of all plays of the semantic game in which II uses τ. The same holds of independence friendly logic of [Hi$_1$96].

Team 5 A team of robots building a car. Each agent s, i.e. robot, has features encoded by the function s. For example, if the domain of the agents is

$$\{\text{reach}, \text{payload}, \text{joint range}, \text{speed}, \text{weight}, \text{paints}, \text{welds}\},$$

we can have a team as in Table 2. Here weight depends on the payload

robot	`reach` (mm)	`payload` (kg)	`range` (°)	`speed` (°/sec)	`weight` (kg)	`paints`	`welds`
1	653	2.5	170	328	28	true	false
2	653	2.5	170	328	28	true	false
3	653	2.5	190	328	28	true	false
4	653	5	170	328	35	true	false
5	653	5	170	300	35	false	true
6	653	5	170	300	35	false	true
7	653	5	360	328	35	false	true

TABLE 2. A team of robots

but not on speed as robots 3 and 4 have the same speed but different weight. Note that the team can be divided into two subteams one of which paints and the other does welding. This team raises the issue whether a team can have two agents with exactly the same values on all features. According to our definition the two agents are in such a case the same. So the above table describing the team is misleading. Robots 1 and 2 are the same, as are robots 5 and 6. If we added the robot number to the domain, the robots would all be different agents. This indicates an extensionality phenomenon in our approach.

3 Dependence types

Our starting point was the idea that teams can manifest dependencies. The different ways that this happens give rise to the concept of a dependence type[3] that we now define. Some dependence types are of an atomic nature in that they are not decomposed further into smaller dependence types. The atomic types are

[3] Types correspond to formulas. We use the word 'type' to emphasize the difference between truth and dependence.

- $t = t'$: Every agent $s \in X$ satisfies $t\langle s\rangle = t'\langle s\rangle$. The team of Table 3 is of type $x_3 = x_4$.
- $\neg t = t'$: Every agent $s \in X$ satisfies $t\langle s\rangle \neq t'\langle s\rangle$. The team of Table 3 is of type $\neg x_2 = x_4$.
- $Rt_1 \ldots t_n$: Every agent $s \in X$ satisfies $(t_1\langle s\rangle, \ldots, t_n\langle s\rangle) \in R^{\mathcal{M}}$. The team of Table 3 is of type $x_2 < x_4$.
- $\neg Rt_1 \ldots t_n$: Every agent $s \in X$ satisfies $(t_1\langle s\rangle, \ldots, t_n\langle s\rangle) \notin R^{\mathcal{M}}$. The team of Table 3 is of type $\neg x_4 < x_2$.
- $=(t_1, \ldots, t_n)$: Every two agents $s, s' \in X$ that satisfy

$$\begin{aligned} t_1\langle s\rangle &= t_1\langle s'\rangle \\ &\vdots \\ t_{n-1}\langle s\rangle &= t_{n-1}\langle s'\rangle, \end{aligned}$$

  also satisfy $t_n\langle s\rangle = t_n\langle s'\rangle$. This type is called the *dependence* type. The team of Table 3 is of type $=(x_1, x_2)$ but not of type $=(x_2, x_1)$. There are two special cases. The first is $=(t)$. This is the type of teams in which the value of t is constant. Team 5 above is of type $=(\texttt{reach})$. The second special case is the type $=()$. This is the type of all teams what so ever and is given a special symbol $\top$.
- $\neg =(t_1, \ldots, t_n)$: This is the type of the empty team $\varnothing$. This should be distinguished from the type of an "impossible" team, that is, the type which is not the type of any team what so ever. We shall return to this type later. The reason for allowing only the empty team to be of the type $\neg =(t_1, \ldots, t_n)$ is the following: We want types to be closed downwards. If a non-empty team was of type $\neg =(t_1, \ldots, t_n)$, there would have to be a singleton team $\{s\}$ of the same type. But singleton teams are always of type $=(t_1, \ldots, t_n)$.

The types $t = t'$, $\neg t = t'$, $Rt_1 \ldots t_n$ and $\neg Rt_1 \ldots t_n$ occur in [Ho$_1$97a] and [Ho$_1$97b]. The dependence type $=(t_1, \ldots, t_n)$ is introduced in [VäHo$_1\infty$]. The empty team is of every atomic type, in particular both of type $t = t'$ and $\neg t = t'$. From atomic dependence types we can build more complex ones with *team operations*, which are the following:

- Negation $\sim\varphi$: The team is not of type φ. The team of Table 3 is of type $\sim =(x_2, x_1)$. Note that $\sim Rt_1 \ldots t_n$ is quite different from $\neg Rt_1 \ldots t_n$. The former says *some* agent s in the team fails to satisfy $(t_1\langle s\rangle, \ldots, t_n\langle s\rangle) \in R^{\mathcal{M}}$, while the latter says *all* fail. Even more

	Domain			
Agent	1	2	3	4
s_1	100	9	10	10
s_2	20	9	10	10
s_3	21	1	10	10
s_4	1	2	11	11
s_5	101	1	11	11

TABLE 3. A team in $(\mathbb{N}, <)$.

	Domain			
Agent	1	2	3	4
s_1	100	9	10	10
s_2	20	9	10	10
s_5	101	1	11	11

	Domain			
Agent	1	2	3	4
s_3	21	1	10	10
s_4	1	2	11	11

TABLE 4. Two teams in $(\mathbb{N}, <)$.

dramatic is the difference between $\sim{=}(t_1, \ldots, t_n)$ and $\neg{=}(t_1, \ldots, t_n)$. The former says some s and s' in the team give the same value to $(t_1, \ldots, t_{n-1})$ but a different value to t_n, while the latter says there are no agents in the team at all. Finally $\sim{=}()$ is the type of no team. Thus we have two negations, $\neg$ and $\sim$, but the former can be applied to atomic types only. Its range of applicability can be extended to what we call $\sim$-free part of team logic.

- Conjunction $\varphi \wedge \psi$: The team is both of type φ and of type ψ. The team of Table 3 is of type $x_3 = x_4 \wedge \neg x_2 = x_4$.

- Tensor $\varphi \otimes \psi$: The team is the union of a team of type φ and a team of type ψ. Team 5 is of type `paints` $\otimes$ `welds`. This indicates the role of $\otimes$ in expressing co-operation skills of teams. The team of Table 3 is of type $=(x_3, x_2) \otimes =(x_3, x_2)$, as Table 4 shows. Note that that team is not of type $=(x_3, x_2)$. So tensor is not idempotent as conjunction is.

- Existential quantifier $\exists x_n \varphi$: The agents $s \in X$ can be modified to $s(a_s/n)$ in such a way that the team $\{s(a_s/n); s \in X\}$ is of type

φ. The team of Table 3 is of type $\exists x_1(x_1 < x_2)$ and also of type $\exists x_5(x_5 < x_2)$.

- Shriek quantifier $!x_n\varphi$: A team X is of this type if the team of all agents $s(a/n)$, where $a \in M$ and $s \in X$, is of type φ. The team of Table 3 is of type $!\,x_1(x_1 < x_3 \otimes x_2 < x_1)$. but not of type $!\,x_1(x_2 < x_1)$.

More formally:

Definition 3.1. Let L be a vocabulary. The set of *types* is defined as follows: If $t, t', t_1, \ldots, t_n$ are terms of the vocabulary L, and R is a relation symbol in L, then the following are types

$$\begin{array}{l} t = t'. \\ \neg t = t'. \\ Rt_1 \ldots t_n. \\ \neg Rt_1 \ldots t_n. \\ =(t_1, \ldots, t_n). \\ \neg =(t_1, \ldots, t_n). \end{array}$$

If φ and ψ are types, then so are:

$$\begin{array}{l} \sim\varphi. \\ \varphi \wedge \psi. \\ \varphi \otimes \psi. \\ \exists x_n\varphi. \\ !x_n\varphi. \end{array}$$

The concept "team X is of type φ" is defined as follows:

- X is of type $t = t'$ iff for all $s \in X(t\langle s\rangle = t'\langle s\rangle)$.
- X is of type $\neg t = t'$ iff for all $s \in X(t\langle s\rangle \neq t'\langle s\rangle)$.
- X is of type $Rt_1 \ldots t_n$ iff for all $s \in X(t_1\langle s\rangle, \ldots, t_n\langle s\rangle) \in R^{\mathcal{M}}$.
- X is of type $\neg Rt_1 \ldots t_n$ iff for all $\in X(t_1\langle s\rangle, \ldots, t_n\langle s\rangle) \notin R^{\mathcal{M}}$.
- X is of type $=(t_1, \ldots, t_n)$ iff for all $s, s' \in X([t_1\langle s\rangle = t_1\langle s'\rangle \& \ldots \& t_{n-1}\langle s\rangle = t_{n-1}\langle s'\rangle]$ implies $t_n\langle s\rangle = t_n\langle s'\rangle)$.
- X is of type $\neg =(t_1, \ldots, t_n)$ iff $X = \varnothing$.
- X is of type $\sim\varphi$ iff X is not of type φ.
- X is of type $\varphi \wedge \psi$ iff X is both of type φ and of type ψ.

- X is of type $\varphi \otimes \psi$ iff $X = Y \cup Z$, where Y is of type φ and Z is of type ψ.
- X is of type $\exists x_n \varphi$ iff there is a function $s \mapsto a_s$ from X to M so that the team $\{s(a_s/n); s \in X\}$ is of type φ.
- X is of type $!x_n \varphi$ iff the team $\{s(a/n) : a \in M, s \in X\}$ is of type φ.

We can define further types from the above ones:

- Disjunction $\varphi \vee \psi$: This is the type $\sim(\sim \varphi \wedge \sim \psi)$, i.e. the type of teams that are of type φ or of type ψ (or both).
- Universal quantifier $\forall x_n \varphi$: This is the type $\sim \exists x_n \sim \varphi$, i.e. the type of teams X such that whenever the agents $s \in X$ are modified to some $s(a_s/n)$, then the new team $\{s(a_s/n); s \in X\}$ is of type φ.
- Implication $\varphi \rightarrow \psi$: This is the type $\sim \varphi \vee \psi$.
- Lollipop $\varphi \multimap \psi$: This is the type $\sim(\varphi \otimes \sim \psi)$, i.e. if whenever it is represented as the union of two teams the first of which is of type φ, then the other one is of type ψ.

There are exactly two teams with empty domain, namely $\varnothing$ and $\{\varnothing\}$. From these we get exactly four different types of teams with empty domain:

Symbol	Type	Teams
$\top$	$=()$	$\varnothing, \{\varnothing\}$
$\bot$	$\sim =()$	
1	$\sim \neg =()$	$\{\varnothing\}$
0	$\neg =()$	$\varnothing$

These *dependence values* form the following diamond:

$$\begin{tikzcd}
 & \top & \\
\mathbf{1} \arrow[ur] & & \mathbf{0} \arrow[ul] \\
 & \bot \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Definition 3.2. Suppose φ is a type of a team with empty domain.[4] We define $\mathcal{M} \models \varphi$ to mean that the team $\{\emptyset\}$ is of type φ in $\mathcal{M}$. We then say $\mathcal{M}$ is *of type* φ.

In a sense, team logic is a four-valued logic. The dependence values of types $\varphi \otimes \psi$, $\varphi \wedge \psi$, and $\sim \varphi$ for types φ and ψ of teams with empty domain can be readily given in terms of truth-tables:

$\otimes$	$\top$	$\bot$	$\mathbf{1}$	$\mathbf{0}$
$\top$	$\top$	$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\mathbf{1}$	$\top$	$\bot$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{0}$	$\top$	$\bot$	$\mathbf{1}$	$\mathbf{0}$

$\wedge$	$\top$	$\bot$	$\mathbf{1}$	$\mathbf{0}$
$\top$	$\top$	$\bot$	$\mathbf{1}$	$\mathbf{1}$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\mathbf{1}$	$\mathbf{1}$	$\bot$	$\mathbf{1}$	$\mathbf{0}$
$\mathbf{0}$	$\mathbf{0}$	$\bot$	$\mathbf{0}$	$\mathbf{0}$

	$\sim$
$\top$	$\bot$
$\bot$	$\top$
$\mathbf{1}$	$\mathbf{0}$
$\mathbf{0}$	$\mathbf{1}$

Here are some examples of types: The type

$$=(x_0, x_1) \otimes =(x_0, x_1)$$

is the type of teams in which field 1 depends on field 0 in the weaker sense than functional dependence that the values of field 0 determine at most two values one of which is the value in field 1. Table 5 is an example of

Rank	Salary
A	2000
A	2100
B	2150
B	2220
C	2340
C	2440
D	2500
D	3100
E	3200
E	3710

TABLE 5. Does salary depend on rank?

a team which is of type $=(\texttt{Rank}, \texttt{Salary}) \otimes =(\texttt{Rank}, \texttt{Salary})$. If we denote $=(x_0, x_1)$ by φ, we get weaker and weaker forms of functional dependence by considering the types

$$\begin{array}{l} \varphi \otimes \varphi \\ \varphi \otimes \varphi \otimes \varphi \\ \varphi \otimes \varphi \otimes \varphi \otimes \varphi \\ \dots \end{array}$$

[4] That is, φ is a 'sentence', it has no free variables.

The type

$$=(x_0, x_1) \wedge =(x_1, x_0)$$

is the type of teams in which field 0 depends functionally on field 1 and conversely field 1 depends functionally on field 0. We could say that fields 0 and 1 are **mutually functionally dependent**. The type

$$\neg x_0 = x_1 \otimes x_2 = x_3$$

is the type of teams in which each agent s which satisfies $s(0) = s(1)$ also satisfies $s(2) = s(3)$. The type

$$\neg x_0 = x_1 \otimes x_2 = x_3$$

is the type of teams in which each agent s which satisfies $s(0) = s(1)$ also satisfies $s(2) = s(3)$. The type

$$(\neg x_0 = x_1 \otimes x_2 = x_3) \wedge (\neg x_2 = x_3 \otimes x_0 = x_1)$$

is the type of teams in which each agent s which satisfies $s(0) = s(1)$ also satisfies $s(2) = s(3)$ and vice versa. The type

$$\begin{aligned} !\, x_0 \exists x_1 \,!\, x_2 \exists x_3 (=(x_2, x_3) \quad & \wedge \quad \neg(x_0 = x_1) \\ & \wedge \quad (\neg x_0 = x_2 \otimes x_1 = x_3) \\ & \wedge \quad (\neg x_1 = x_2 \otimes x_3 = x_0)) \end{aligned}$$

is the type of teams which are non-empty if and only the underlying set M is either infinite or finite and of even cardinality. The type

$$\begin{aligned} !\, x_0 \exists x_1 \,!\, x_2 \exists x_3 (=(x_2, x_3) \quad \wedge \quad & (\neg P x_0 \otimes (Q x_1 \wedge \\ & (\neg x_0 = x_2 \otimes x_1 = x_3) \\ & (\neg x_1 = x_3 \otimes x_0 = x_2)))) \end{aligned}$$

is the type of teams which are non-empty if and only if in the underlying structure $\mathcal{M}$ the predicate P and Q satisfy $|P^{\mathcal{M}}| \leq |Q^{\mathcal{M}}|$.

4 Modes of dependence and independence

We shall now use the above concept of dependence type to analyze in more general terms different concepts of dependence and independence.

Let X be a team with domain $\{m_1, \ldots, m_n\}$ in a domain M, as in Table 1. There is an obvious partial order in the powerset of $\{m_1, \ldots, m_n\}$, namely the set-theoretical subset-relation $\subseteq$. We now define a new relation, called the pre-order of functional dependence:

$V \leq W$	V is *functionally dependent* on W, i.e. features in V can be determined if the values of the features in W are known. In symbols, $\forall s, s' \in X((\forall y \in W(s(y) = s'(y)) \to (\forall x \in V(s(x) = s'(x)))))$. Equivalently: V is functionally dependent on $\{w_1, \ldots, w_n\}$, if for all $y \in V$ there is a function f_y such that for all s in X: $s(y) = f_y(s(w_1), \ldots, s(w_n))$.

It is evident from the definition that the pre-order of functional dependence is weaker than the partial order of inclusion in the sense that every subset of a set obviously depends functionally on the set itself. It is more interesting that sometimes a set is functionally dependent on a set disjoint from itself, and a singleton set may be functionally dependent of another singleton set. Some sets may be functionally dependent on the empty set (in that case the feature has to have a constant value). Every set is certainly functionally dependent on the whole universe.

Note that the Armstrong Axioms of functional dependence (see [Ar74]) state exactly the following:

1. $V \leq W$ is a pre-order, i.e. reflexive and transitive.
2. If $V \subseteq W$, then $V \leq W$.
3. If $V \leq W$ and U is arbitrary, then $V \cup U \leq W \cup U$.
4. If $V \leq W$, then there is a minimal $U \subseteq W$ such that $V \leq U$.

These axioms characterize completely when a functional dependence $V \leq W$ follows from given functional dependencies $V_1 \leq W_1, \ldots, V_n \leq W_n$.

Now we can define two versions of dependence, by using the pre-order of determination. Suppose for $W \cap V = \varnothing$. We define:

V is **dependent** on W	There is some minimal $U \geq V$ such that $U \cap V = \varnothing$ and $W \subseteq U$.
V is **totally dependent** on W	For every $U \geq V$ such that $U \cap V = \varnothing$ we have $W \subseteq U$.
V is **independent** of W	There is some $U \geq V$ such that $U \cap V = \varnothing$ and $W \cap U = \varnothing$.
V is **totally independent** of W	For every minimal $U \geq V$ such that $U \cap V = \varnothing$ we have $W \cap U = \varnothing$.
V is **non-determined**	There is no $U \geq V$ such that $U \cap V = \varnothing$. In the opposite case V is called determined.

`star`	`food`	`drink`	`music`
Marlon	pasta	wine	classical
Jack	pasta	beer	classical
Robert	steak	wine	rock
Julia	steak	beer	rock

TABLE 6. Team of favorites.

Note, that it is quite conceivable that two sets V and W of features are *mutually dependent* in the sense that both depend on each other. In the team of Table 6 we can make the following observations: `star` depends on `food`, since $\{\texttt{star}\}\leq\{\texttt{food}, \texttt{drink}\}$. On the other hand, `star` depends also on `music`, since $\{\texttt{star}\}\leq\{\texttt{drink}, \texttt{music}\}$. So `star` is not totally dependent on either `food` or `music` but it is totally dependent on `drink`.

Note that the above concepts are defined with respect to the fields that we have in the domain. Indeed, it seems meaningless to define what independence means in a domain where any *new* fields can be introduced. The new fields can change independence to dependence completely.

Here are some immediate relationships between the introduced concepts of dependence and independence:

1. Every V is totally dependent and totally independent on $\varnothing$.
2. If V is (totally) dependent on W, then V is (totally) dependent on every subset of W.
3. If V is dependent on W, it can still be also independent of W, but not totally, unless $W = \varnothing$.
4. V is independent of $\{x\}$ if and only if V is not totally dependent on $\{x\}$.
5. V is totally independent of $\{x\}$ if and only if V is not dependent on $\{x\}$.

If $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_m\}$, let $=(U, V)$ be the type

$$\bigwedge_{i=1}^{m} =(u_1, \ldots, u_n, v_i).$$

This is the type of teams in which V is functionally dependent on U. It is clear that we can define dependence, independence total dependence, total independence, and non-determinedness in terms of the basic types $=(U, V)$. We suggest that the types of team logic provide a proper framework for an analysis of the variety of different concepts related to dependence.

5 Team algebra

Let D be a fixed domain and $\mathcal{T}$ the set of all teams with domain D. Let us write $\varphi \Leftrightarrow \psi$ if every team of type φ is of type ψ and vice versa. We call this relation *logical equivalence*. The Boolean operations $\wedge, \vee, \sim$ together with $\top$ and $\bot$ obey usual laws of Boolean algebras, and the mapping

$$h(\varphi) = \{X \in \mathcal{T} : X \text{ is of type } \varphi\}.$$

is a homomorphism between dependence types, endowed with the operations $\wedge, \vee$ and $\sim$, and the Boolean algebra of subsets of $\mathcal{T}$:

$$\begin{array}{lcl} h(\varphi \wedge \psi) & = & h(\varphi) \cap h(\psi) \\ h(\varphi \vee \psi) & = & h(\varphi) \cup h(\psi) \\ h(\sim \varphi) & = & \mathcal{T} - h(\varphi) \\ h(\top) & = & \mathcal{T} \\ h(\bot) & = & \varnothing \end{array}$$

The operation $\otimes$ satisfies the laws

$$\begin{array}{lcl} \varphi \otimes \psi & \Leftrightarrow & \psi \otimes \varphi \\ \varphi \otimes (\psi \otimes \theta) & \Leftrightarrow & (\varphi \otimes \psi) \otimes \theta \\ \varphi \otimes (\psi \vee \theta) & \Leftrightarrow & (\varphi \otimes \psi) \vee (\varphi \otimes \theta) \\ \varphi \otimes \bot & \Leftrightarrow & \bot \\ \varphi \otimes \mathbf{0} & \Leftrightarrow & \varphi \\ \mathbf{1} \otimes \mathbf{1} & \Leftrightarrow & \mathbf{1} \\ \top \otimes \top & \Leftrightarrow & \top \end{array}$$

but

$$\begin{array}{lcl} \varphi \otimes (\psi \wedge \theta) & \not\Leftrightarrow & (\varphi \otimes \psi) \wedge (\varphi \otimes \theta) \\ \varphi \wedge (\psi \otimes \theta) & \not\Leftrightarrow & (\varphi \wedge \psi) \otimes (\varphi \wedge \theta) \\ \varphi \vee (\psi \otimes \theta) & \not\Leftrightarrow & (\varphi \vee \psi) \otimes (\varphi \vee \theta) \\ \varphi \otimes \varphi & \not\Leftrightarrow & \varphi \end{array}$$

For quantifiers we have

$$\begin{array}{lcl} \forall x_n(\varphi \wedge \psi) & \Leftrightarrow & \forall x_n \varphi \wedge \forall x_n \psi \\ \forall x_n \forall x_m \varphi & \Leftrightarrow & \forall x_m \forall x_n \varphi \\ \exists x_n(\varphi \vee \psi) & \Leftrightarrow & \exists x_n \varphi \vee \exists x_n \psi \\ \exists x_n \exists x_m \varphi & \Leftrightarrow & \exists x_m \exists x_n \varphi \end{array}$$

The shriek ! commutes with every team operation except $\exists$:

$$\begin{array}{lcl} !x_n \sim \varphi & \Leftrightarrow & \sim !x_n \varphi \\ !\, x_n(\varphi \otimes \psi) & \Leftrightarrow & !\, x_n \varphi \otimes !\, x_n \psi \\ !\, x_n(\varphi \wedge \psi) & \Leftrightarrow & !\, x_n \varphi \wedge !\, x_n \psi \\ !\, x_n(\varphi \vee \psi) & \Leftrightarrow & !\, x_n \varphi \vee !\, x_n \psi \\ !\, x_n \forall x_m \varphi & \Leftrightarrow & \forall x_m \, !\, x_n \varphi \\ !\, x_n \exists x_m \varphi & \not\Leftrightarrow & \exists x_m \, !\, x_n \varphi \end{array}$$

Obviously, there is a lot more one can say about team algebra.

6 Some translations

We show that the $\sim$-free fragment of team logic is equivalent to independence friendly logic. To make our result exact we choose what we believe is the best behaving version of independence friendly logic. This is the *dependence friendly logic* in which the quantifier

$$\exists x_n \backslash t_1 \dots t_m \varphi$$

has the meaning "there is a value for x_n, functionally depending on what the values of $t_1, \dots, t_m$ are, such that φ". In the original independence friendly logic the quantifier

$$\exists x_n / \forall x_1 \dots \forall x_m \varphi$$

had the meaning "there is a value for x_n, independently of what the values of $x_1, \dots, x_m$ are, such that φ". An attempt to understand what "independently" might mean here led the author to the considerations of Section 4 and eventually to the more basic concept of functional dependence.

We can easily define a translation $\varphi \mapsto \varphi^*$ of dependence friendly logic into team logic, but we have to assume that the formula φ of dependence friendly logic is in negation normal form:

$$\begin{array}{lcl}
(t = t')^* & = & t = t' \\
(\neg t = t')^* & = & \neg t = t' \\
(Rt_1 \dots t_n)^* & = & Rt_1 \dots t_n \\
(\neg Rt_1 \dots t_n)^* & = & \neg Rt_1 \dots t_n \\
(\varphi \vee \psi)^* & = & \varphi^* \otimes \psi^* \\
(\varphi \wedge \psi)^* & = & \varphi^* \wedge \psi^* \\
(\exists x_n \backslash t_1, \dots, t_n \varphi)^* & = & \exists x_n (=(t_1, \dots, t_n, x_n) \wedge \varphi^*) \\
(\forall x_n \varphi)^* & = & !x_n \varphi^*
\end{array}$$

It is an immediate consequence of the definitions that for all $\mathcal{M}$, all φ, and all X we have

$$\mathcal{M} \models_X \varphi \text{ in dependence friendly logic if and only if } X \text{ is of type } \varphi^*.$$

So we may consider dependence friendly logic a fragment of team logic, and team logic an extension of dependence friendly logic obtained by adding classical negation.

Conversely, we can define a translation $\varphi \mapsto \varphi^+$ of the $\sim$-free fragment of team logic into dependence friendly logic:

$$\begin{array}{lcl}
(t = t')^+ & = & t = t' \\
(\neg t = t')^+ & = & \neg t = t' \\
(Rt_1 \dots t_n)^+ & = & Rt_1 \dots t_n \\
(\neg Rt_1 \dots t_n)^+ & = & \neg Rt_1 \dots t_n \\
(=(t_1, \dots, t_n))^+ & = & \exists x_m \backslash t_1, \dots, t_{n-1}(t_n = x_m), \text{ where} \\
 & & x_m \text{ is not free in } \varphi \\
(\neg =(t_1, \dots, t_n))^+ & = & \exists x_0(\neg x_0 = x_0) \\
(\varphi \otimes \psi)^+ & = & \varphi^+ \vee \psi^+ \\
(\varphi \wedge \psi)^+ & = & \varphi^+ \wedge \psi^+ \\
(\exists x_n \varphi)^+ & = & \exists x_n \backslash x_{m_1}, \dots, x_{m_k} \varphi^+, \text{ where } x_{m_1}, \dots, x_{m_k} \\
 & & \text{are the free variables of } \varphi \text{ other than } x_n \\
(!\, x_n \varphi)^+ & = & \forall x_n \varphi^+
\end{array}$$

It is again an immediate consequence of the definitions that for all $\mathcal{M}$, all φ, and all X we have

X is of type φ if and only if $\mathcal{M} \models_X \varphi^+$ in dependence friendly logic.

The two translations $\varphi \mapsto \varphi^*$ and $\varphi \mapsto \varphi^+$ demonstrate clearly that team logic is built on top of dependence friendly logic, and a fortiori, on top of independence friendly logic. The addition team logic brings to independence friendly logic is classical negation. As this paper shows this addition calls for rather substantial revision of independence friendly logic.

It is well-known[5] that independence friendly logic can be presented in the existential fragment Σ^1_1 of second order logic. If the same presentation is applied to team logic, we go up from Σ^1_1 to the unrestricted second order logic. The proof of the following theorem is an easy adaption of the corresponding result for independence friendly logic in [Ho$_1$97a]:

Theorem 6.1. We can associate with every type $\varphi(x_{i_1}, \dots, x_{i_n})$ in vocabulary L a second order sentence $\eta_\varphi(U)$, where U is n-ary, such that for all L-structures $\mathcal{M}$ and teams X with $\mathrm{dom}(X) = \{i_1, \dots, i_n\}$ the following conditions are equivalent

1. X is of type φ in $\mathcal{M}$.
2. $(\mathcal{M}, X) \models \eta_\varphi(U)$.

Corollary 6.2. For every type φ there is a second order sentence η_φ such that for all models $\mathcal{M}$ we have $\mathcal{M} \models \varphi$ if and only if $\mathcal{M} \models \eta_\varphi$.

[5] For details, we refer to [Ho$_1$97a].

With the translation $\varphi \mapsto \eta_\varphi$ we can consider team logic a fragment of second order logic, even if the origin of team logic in the dependence relation $=(t_1, \ldots, t_n)$ is totally different from the origin of second order logic, and even if the logical operations of team logic are totally different from those of second order logic.

We could draw many immediate conclusions from Corollary 6.2 and from what is known about second order logic.

We have given a translation of team logic into second order logic. Now we give an implicit translation of second and higher order logic in team logic. The translation is implicit in the sense that it uses new predicates and an extension of the universe. However, the new predicates and the new universe are unique up to isomorphism. We omit the quite standard proof of the following result, which is essentially contained already in [En70] and [Kr$_2$La$_0$79].

Theorem 6.3. Suppose L is a vocabulary and $n \in \mathbb{N}$. There is a type φ in the vocabulary $L' = L \cup \{P, E\}$ such that for all L-structures $\mathcal{M}$ there is a unique (mod $\cong$) $\mathcal{N}$ of type φ with $(\mathcal{N} \restriction L)^{(P^{\mathcal{N}})} = \mathcal{M}$. Moreover, we can associate with every sentence ψ of second order logic in vocabulary L, with no second order variables of arity $> n$, a dependence type ξ_ψ in the vocabulary L' such that the following conditions are equivalent for all L-structures $\mathcal{M}$:

1. $\mathcal{M} \models \psi$
2. $\mathcal{N} \models \xi_\psi$ for the unique (mod $\cong$) $\mathcal{N}$ such that $\mathcal{N} \models \varphi$ and $(\mathcal{N} \restriction L)^{(P^{\mathcal{N}})} = \mathcal{M}$.

Corollary 6.4. A second order sentence ψ has a model if and only if $\varphi \wedge \xi_\psi$ is the type of $\{\varnothing\}$ in some model.

With the translation $\varphi \mapsto \xi_\varphi$ we can consider second order logic an implicitly defined fragment of team logic. An explicit translation, following Harel [Ha$_1$79], has been constructed by Ville Nurmi.

The *decision problem* of a logic is the problem, with a sentence φ as input, whether $\mathcal{M} \models \varphi$ for all $\mathcal{M}$. The *consistency problem* of a logic is the problem, with a sentence φ as input, whether $\mathcal{M} \models \varphi$ for some $\mathcal{M}$. The *Löwenheim number* of a logic is the smallest cardinal κ such that for any sentence φ, if $\mathcal{M} \models \varphi$ for some $\mathcal{M}$, then $\mathcal{M} \models \varphi$ for some $\mathcal{M}$ of cardinality $\leq \kappa$. The *Hanf number* of a logic is the smallest cardinal κ such that for any sentence φ, if $\mathcal{M} \models \varphi$ for some $\mathcal{M}$ of cardinality $\geq \kappa$, then $\mathcal{M} \models \varphi$ for models $\mathcal{M}$ of arbitrarily large cardinality. A logic satisfies the Suslin-Kleene Interpolation Theorem if every model class which is a relativized reduct of a definable model class and whose complement has the same property is itself a definable model class. The Δ-extension of a logic is the smallest extension of it to a logic with the Suslin-Kleene Interpolation Theorem. For more details concerning these concepts we refer to [Ba$_6$Fe$_0$85].

Corollary 6.5. The decision problems of team logic and second order logic are recursively isomorphic (and Π_2-complete, see [Vä01] and Footnote 6). They have the same Löwenheim and Hanf numbers. They have the same Δ-extension.

Note that the corresponding corollary for independence friendly logic says: The decision problems of independence friendly logic and second order logic are recursively isomorphic. The consistency problem of independence friendly logic is co-r.e. while that of second order logic is Σ_2-complete[6]. The Löwenheim and Hanf number of independence friendly logic is $\aleph_0$, while the Löwenheim number of second order logic is between the first measurable and the first supercompact cardinals, if such exist, and the Hanf number of second order logic is between the first supercompact and the first extendible cardinal, if such exist (see [Ma$_0$71]).

7 Ehrenfeucht-Fraïssé game

We now introduce an Ehrenfeucht-Fraïssé game adequate for team logic and use this game to characterize team logic. This game is nothing else than a "two-directional" version of the Ehrenfeucht-Fraïssé game of independence friendly logic presented in [Vä02].

Definition 7.1. Let $\mathcal{M}$ and $\mathcal{N}$ be two structures of the same vocabulary. The game $\mathrm{EF}_n^{\mathrm{TL}}$ has two players and n moves. The position after move m is a pair (X, Y), where $X \subseteq M^{i_m}$ and $Y \subseteq N^{i_m}$ for some i_m. In the beginning the position is $(\varnothing, \varnothing)$ and $i_0 = 0$. Suppose the position after move number m is (X, Y). There are the following possibilities for the continuation of the game:

Splitting move: Player I represents X (or Y) as a union $X = X_0 \cup X_1$. Then player II represents Y (respectively, X) as a union $Y = Y_0 \cup Y_1$. Now player I chooses whether the game continues from the position (X_0, Y_0) or from the position (X_1, Y_1).

[6] We use the notation of Lévy [Lé65]. The Σ_0-formulas, which are at the same time called Π_0-formulas, are all formulas in the vocabulary $\{\in\}$ obtained from atomic formulas by the operations $\neg, \vee, \wedge$ and the *bounded quantifiers* $\exists x_0(x_0 \in x_1 \wedge \varphi)$ and $\forall x_0(x_0 \in x_1 \to \varphi)$. The Σ_{n+1}-formulas are obtained from Π_n-formulas by existential quantification. The Π_{n+1}-formulas are obtained from Σ_n-formulas by existential quantification. A set $P \subseteq \mathbb{N}$ is called Σ_n-definable if there is a Σ_n-formula $\varphi(x_0)$ of set theory such that $n \in P \iff \varphi(n)$. A problem is called Σ_n-complete if the set itself is Σ_n-definable and, moreover, for every Σ_n-definable set $X \subseteq \mathbb{N}$ there is a recursive function $f : \mathbb{N} \to \mathbb{N}$ such that $n \in X \iff f(n) \in P$. The concepts of a Π_n-definable set and a Π_n-complete set are defined analogously.

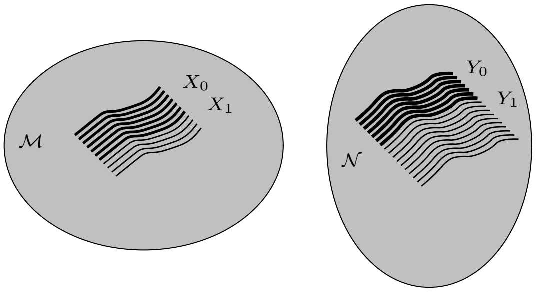

FIGURE 3. A splitting move

Duplication move: Player I decides that the game should continue from the new position

$$(X(M/i_m), Y(N/i_m)).$$

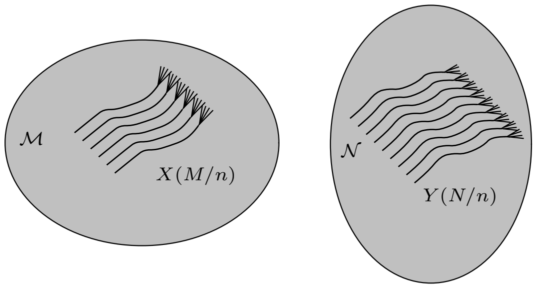

FIGURE 4. A duplication move

Supplementing move: Player I chooses a function $F : X \to M$ (or $F : Y \to N$). Then player II chooses a function $G : Y \to N$ (respectively, $G : X \to M$). Then the game continues from the position $(X(F/i_m), Y(G/i_m))$.

After n moves the position (X_n, Y_n) is reached and the game ends. Player II is the winner, if

$$X_n \text{ is of type } \varphi \text{ in } \mathcal{M} \Leftrightarrow Y_n \text{ is of type } \varphi \text{ in } \mathcal{N}$$

holds for all atomic types $\varphi(x_0, \ldots, x_{i_n-1})$. Otherwise player I wins.

This is a game of perfect information and the concept of winning strategy is defined as usual. The game is determined by the Gale-Stewart theorem.

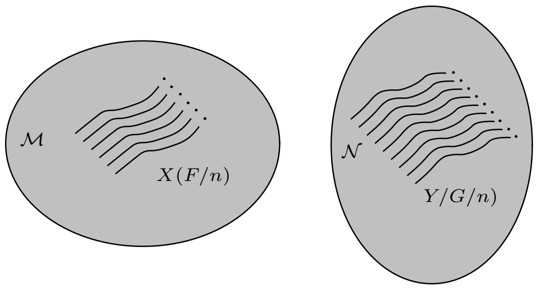

FIGURE 5. A supplementing move

Define

$$\begin{array}{lcl}
\mathrm{qr}^{\mathrm{TL}}(\varphi) & = & 0 \text{ if } \varphi \text{ is atomic,}\\
\mathrm{qr}^{\mathrm{TL}}(\varphi \otimes \psi) & = & \max(\mathrm{qr}^{\mathrm{TL}}(\varphi), \mathrm{qr}^{\mathrm{TL}}(\psi)){+}1,\\
\mathrm{qr}^{\mathrm{TL}}(\varphi \wedge \psi) & = & \max(\mathrm{qr}^{\mathrm{TL}}(\varphi), \mathrm{qr}^{\mathrm{TL}}(\psi)),\\
\mathrm{qr}^{\mathrm{TL}}(\exists x_n \varphi) & = & \mathrm{qr}^{\mathrm{TL}}(\varphi){+}1,\\
\mathrm{qr}^{\mathrm{TL}}(!x_n \varphi) & = & \mathrm{qr}^{\mathrm{TL}}(\varphi){+}1,\\
\mathrm{qr}^{\mathrm{TL}}(\sim \varphi) & = & \mathrm{qr}^{\mathrm{TL}}(\varphi).
\end{array}$$

Let Type_n^m be the set of types φ with $\mathrm{qr}^{\mathrm{TL}}(\varphi) \leq m$ and with free variables among $x_0, \ldots, x_{n-1}$. We write $\mathcal{M} \equiv_{\mathrm{TL}}^n \mathcal{N}$, if $\mathcal{M} \models \varphi$ is equivalent to $\mathcal{N} \models \varphi$ for all φ in Type_0^n, and $\mathcal{M} \equiv_{\mathrm{TL}} \mathcal{N}$ if $\mathcal{M} \equiv_{\mathrm{TL}}^n \mathcal{N}$ for all n. Note that there are for each n and m, up to logical equivalence, only finitely many types in Type_n^m.

Theorem 7.2. Suppose $\mathcal{M}$ and $\mathcal{N}$ are models of the same vocabulary. Then the following conditions are equivalent:

(1) Player II has a winning strategy in the game $\mathrm{EF}_n^{\mathrm{TL}}(\mathcal{M}, \mathcal{N})$.

(2) $\mathcal{M} \equiv_{\mathrm{TL}}^n \mathcal{N}$.

Proof. It is easy to prove by induction on m the equivalence, for all n, of the following two statements:

$(3)_m$ Player II has a winning strategy in the game $\mathrm{EF}_m^{\mathrm{TL}}(\mathcal{M}, \mathcal{N})$ in position (X, Y), where $X \subseteq M^n$ and $Y \subseteq N^n$.

$(4)_m$ If φ is a type in Type_n^m, then X is of type φ in $\mathcal{M}$ if and only if Y is of type φ in $\mathcal{N}$.

Q.E.D.

Corollary 7.3. Suppose $\mathcal{M}$ and $\mathcal{N}$ are models of the same vocabulary. Then the following conditions are equivalent:

(1) $\mathcal{M} \equiv_{\mathrm{TL}} \mathcal{N}$.

(2) For all natural numbers n, player II has a winning strategy in the game $\mathrm{EF}_n^{\mathrm{TL}}(\mathcal{M}, \mathcal{N})$.

Using the fact that there are for each n and m, up to logical equivalence, only finitely many types in Type_n^m, it is easy to prove that a model class K is the class of models of a type in Type_0^n if and only if K is closed under the relation $\equiv_{\mathrm{DF}}^n$.

From this and the above theorem we get:

Corollary 7.4. Suppose K is a model class. Then the following conditions are equivalent:

(1) K is the class of models of a type of team logic.

(2) There is a natural number n such that K is closed under the relation

$$\mathcal{M} R \mathcal{N} \iff \text{Player } II \text{ has a winning strategy in } \mathrm{EF}_n^{\mathrm{TL}}(\mathcal{M}, \mathcal{N}).$$

We have obtained, after all, a purely game-theoretic definition of team logic.

References

[Ar74] W.W. Armstrong. Dependency structures of data base relationships. *Information Processing* 74:580–583, 1974.

[$\mathrm{Ba_6Fe_0}$85] J. Barwise & S. Feferman, eds. *Model-theoretic logics.* Springer, 1985.

[En70] H.B. Enderton. Finite partially-ordered quantifiers. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* 16:393–397, 1970.

[$\mathrm{Ha_1}$79] D. Harel. Characterizing second-order logic with first-order quantifiers. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* 25(5):419–422, 1979.

[$\mathrm{Hi_1}$96] J. Hintikka. *The Principles of Mathematics Revisited.* Cambridge University Press, 1996.

[$\mathrm{Ho_1}$97a] W. Hodges. Compositional semantics for a language of imperfect information. *Logic Journal of the IGPL* 5(4):539–563, 1997.

[$\mathrm{Ho}_1$97b] W. Hodges. Some strange quantifiers. In [$\mathrm{MyRo}_4\mathrm{Sa}_1$97, pp. 51–65].

[$\mathrm{Kr}_2\mathrm{La}_0$79] M. Krynicki & A.H. Lachlan. On the semantics of the Henkin quantifier. *Journal of Symbolic Logic* 44(2):184–200, 1979.

[Lé65] A. Lévy. A hierarchy of formulas in set theory. *Memoirs of the American Mathematical Society* 57:76, 1965.

[$\mathrm{Ma}_0$71] M. Magidor. On the role of supercompact and extendible cardinals in logic. *Israel Journal of Mathematics* 10:147–157, 1971.

[$\mathrm{MyRo}_4\mathrm{Sa}_1$97] J. Mycielski, G. Rozenberg & A. Salomaa, eds. *Structures in Logic and Computer Science, A Selection of Essays in Honor of Andrzej Ehrenfeucht*, *Lecture Notes in Computer Science* 1261. Springer, 1997.

[Vä01] J. Väänänen. Second-order logic and foundations of mathematics. *The Bulletin of Symbolic Logic* 7(4):504–520, 2001.

[Vä02] J. Väänänen. On the semantics of informational independence. *Logic Journal of the IGPL* 10(3):339–352, 2002.

[Vä07] J. Väänänen. *Dependence Logic: A New Approach to Independence Friendly Logic*, *London Mathematical Society Student Texts* 70. Cambridge University Press, 2007.

[$\mathrm{VäHo}_1\infty$] J. Väänänen & W. Hodges. Dependence of variables construed as an atomic formula. Preprint.