Team building in dependence

Julian Bradfield

Laboratory for Foundations of Computer Science, University of Edinburgh,
10 Crichton St, Edinburgh, EH8 9AB, U.K.
jcb@inf.ed.ac.uk

Abstract

Hintikka and Sandu's Independence-Friendly Logic was introduced as a logic for partially ordered quantification, in which the independence of (existential) quantifiers from previous (universal) quantifiers is written by explicit syntax. It was originally given a semantics by games of imperfect information; Hodges then gave a (necessarily) second-order Tarskian semantics. More recently, Väänänen (2007) has proposed that the many curious features of IF logic can be better understood in his Dependence Logic, in which the (in)dependence of variables is stated in atomic formula, rather than by changing the definition of quantifier; he gives semantics in Tarskian form, via imperfect information games, and via a routine second-order perfect information game. He then defines Team Logic, where classical negation is added to the mix, resulting in a full second-order expressive logic. He remarks that no game semantics appears possible (other than by playing at second order). In this article, we explore an alternative approach to game semantics for DL, where we avoid imperfect information, yet stay locally apparently first-order, by sweeping the second-order information into longer games (infinite games in the case of countable models). Extending the game to Team Logic is not possible in standard games, but we conjecture a move to transfinite games may achieve a 'natural' game for Team Logic.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases partially ordered quantification, independence-friendly logic, game semantics

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.116

1 Introduction

In the 60s, Henkin [9] introduced partially ordered quantifiers, which extend first-order logic (FOL) by quantifiers such as $\begin{smallmatrix}\forall x & \exists y \\ \forall u & \exists v\end{smallmatrix}$, where the existential choice of y is to be made without knowing the value of u, and similarly v is chosen independently of x. Subsequent work [8, 17] establish some basic properties, such as the equi-expressiveness of such quantifiers with existential second-order logic (ESOL), but little more was done. Then Hintikka and Sandu [10] gave new life to the topic with a provocative paper, which introduced a new linear syntax with the independence explicitly noted ($\forall x.\forall u.\exists y/u.\exists v/x$), gave it a semantics by extending Hintikka's celebrated games for FOL to be games of imperfect information, so that truth and falsity are defined to be the existence of a winning strategy one or other of the two players, demonstrated a number of surprising properties of the logic, and argued that it should displace FOL as the foundation of mathematics. Although the last claim has been generally politely ignored, a community has grown up exploring the ramifications of independence, and the properties of Independence-Friendly Logic (IFL) have continued to surprise us – the recent textbook [13] provides a comprehensive survey of the mainstream of IFL research. Branching off from the IFL river are a number of tributary streams, including logics for agent-based systems [14], and work by myself and colleagues on the relationship of IFL to logics for concurrency [6, 4]. One extension of IFL studied by Kreutzer and myself

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 116–128

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

[5] was the IF version of Least Fixpoint Logic (IF-LFP), in which the least fixpoint operator on relations is added. At that time, we were able to show, by roundabout means, that this extension is very powerful, and on finite models includes all of SOL. Our semantics there was the fixpoint extension of Hodges' [11] Tarskian semantics for IFL, a semantics which is second-order in nature, as it must be to describe the power of IF.

More recently, Väänänen has been developing the thesis that IFL is perhaps not the best notation for understanding the problems posed by independent quantification. This work is collected, up to 2007, in the textbook [16]. Väänänen proposes that, instead of annotating quantifiers with (in)dependence requirements, one should leave them alone, and instead use, as atomic formulae, *dependence atoms* of the form $\mathscr{D}(\vec{x}, y)$, which means that the value of y is functional in the values of $\vec{x}$ (and is therefore independent of any other variables that may be in scope). So instead of $\forall x.\forall u.\exists y/u.\exists v/x.\ldots$, we write $\forall x.\forall u.\exists y.\exists v.\mathscr{D}(x,y) \wedge \mathscr{D}(u,v) \wedge \ldots$. This he calls Dependence Logic (DL). The primary semantics for DL is a Tarskian semantics, necessarily second-order, which is essentially the semantics given by Hodges for IFL; however, as he does not need to deal with the slashed quantifiers of IFL, the semantics is considerably easier to use in proving results about DL (such as the extremely high undecidability of its validity problem). There are two derived semantics: an imperfect information game which reconstructs the Hintikka game, though it appears less natural, and a perfect information game, which of course has second order moves and is basically the standard Hintikka SOL game for the semantics.

One of the bugbears throughout the history of research on IF and its descendants is negation. There are two ways of understanding negation: in the game-theoretic understanding of IFL, it is natural to think of negation via the game-theoretic duality familiar from the FOL game: negation corresponds to swapping the roles of the two players. However, this is not the same as negation in the classical sense. As the IF games are of imperfect information, they are not determined, and so a formula may be neither true nor false in the usual understanding (for example, $\forall x.\exists y/x.x = y$ is not true on a 2-element domain, but neither is its dual $\exists x.\forall y/x.x \neq y$). Hintikka considered forming the boolean closure of IFL, and Hodges [11] introduced a technique for dealing with classical negation in a limited sense.

True classical negation corresponds to saying (in the game view) that a player 'does not have a winning strategy'. It is therefore apparently a firmly second order operator. In the DL framework, Väänänen's primary semantics is already second-order, and it is therefore immediate to add classical negation: a team satisfies 'not ϕ' if it does not satisfy ϕ. The result is a logic with full second order power, and the ability easily to say highly complex properties.

Väänänen considered that as Team Logic expresses SOL, it was hard to imagine a game theoretic semantics (other than the trivial semantics by playing games directly at second order in the meta-language). In this article, we introduce a novel game semantics for DL, and discuss how one can imagine extending it to TL, although imagination has yet to be made real. Of course, such games have very high complexity to solve, but they fit conceptually within the game semantics of IF logic.

We will start the paper by introducing, completely but very concisely, the notations and concepts of IFL and DL, and also explain the intuitions which underly them – a long-standing feature of IF research has been that it is truly philosophical logic, not just formal symbol shuffling. We will then describe the intuition behind our alternative game semantics, and then proceed to its formal definition and proof of correctness.

2 Preliminaries

Notation and terminology is not entirely standardized. Where there are several options, I shall generally follow Väänänen, but sometimes Hodges. I shall use standard concepts such as free and bound variables without further explanation. Formulae are always considered as abstract syntax trees, and words such as 'earlier' and 'above' should be so interpreted.

An important convention is that 'subformula' always means 'node in the abstract syntax tree', and includes the formula itself; unlike FOL, in IF logics one may need to treat different occurrences of the same textual subformula differently.

Throughout we assume some structure $\mathscr{M}$ with a domain M of values which may be bound to variables of the logics. All the logics may be defined to include constants, function symbols and terms built thereby, and interpreted by operations in $\mathscr{M}$, but to reduce notation, we shall generally state definitions without them.

We will start by, as Hintikka did originally, ignoring negation, and working entirely in negation normal form with both $\wedge$ and $\vee$, and both $\forall$ and $\exists$. After introducing IF and DL in this form, we will discuss negation.

2.1 IF logic syntax and semantics

The syntax is that of FOL in negation normal form (where, as usual, we shall let $x, y, \dots \in Var$ denote variables, and $P, Q, \dots \in Prop$ denote relational atoms) – we write $\not P$ for negated atoms. The equality symbol is usually included, but for our purposes may be treated as a relational atom rather than a distinct primitive. In addition, there are 'slashed quantifiers' $\exists x/\vec{y}$ and $\forall x/\vec{y}$, where $\vec{y}$ is a subset of the variables bound earlier in the formula (or, in general, of those variables and the free variables of the formula). '$/\vec{y}$' does not bind the $\vec{y}$.

The semantics of a sentence ϕ (the original presentation does not account for open formulae) is given by a game between two players, Eloise and Abelard. The game is played just as for the standard Hintikka game for FOL: at $\exists x.\psi$ ($\forall x.\psi$), Eloise (Abelard) chooses a value for x; at $\psi_0 \vee \psi_1$ ($\psi_0 \wedge \psi_1$), Eloise (Abelard) chooses to proceed to ψ_0 or ψ_1. At atoms and negated atoms, Eloise (Abelard) wins iff the formula is true (false) with the current binding of variables to values.

At a slashed quantifier $\exists x/\vec{y}$ ($\forall x/\vec{y}$), Eloise (Abelard) is required to make her (his) choice of y without knowing the currently bound values of $\vec{y}$.

ϕ is said to be *true* iff Eloise has a winning strategy in this game of imperfect information; *false* iff Abelard does; and *undetermined* otherwise.

An alternative, and more mathematically tractable, way of phrasing the semantics is to drop any reference to 'imperfect information' in the moves of the game, and instead say that the players are subject to constraints in the strategies they are allowed to use: their strategies at slashed quantifiers must be *uniform* in the $\vec{y}$.

2.2 DL syntax and semantics

DL has the syntax of FOL (in negation normal form for the present), with addition of *dependence atoms* $\mathscr{D}(\vec{x}, y)$ and negated dependence atoms $\not\mathscr{D}(\vec{x}, y)$

An *assignment* s is a mapping from a set $\mathrm{dom}(s)$ of variables to values in M. $s(v/y)$ denotes the assignment s extended (or updated) to map y to v.

A *team* is a set of assignments with a common domain. If X is a team, $X(M/y)$ denotes the team $\{s(v/y) : s \in X, v \in M\}$, i.e. X extended by all possible choices for y. If $F\colon X \to M$

is a function choosing one value for each member of X, $X(F/y)$ denotes the team $\{s(F(s)/y) : s \in X\}$, i.e. X extended by one choice of y for each member of the team.

A *triple* is a tuple (ϕ, X, d) where d is 0 or 1. The intended interpretation is that $(\phi, X, 1)$ means that the team X makes the formula ϕ true, and $(\phi, X, 0)$ means that X makes ϕ false – as with IF logic, it may be that neither holds.

The semantics of DL is given by defining the set $\mathscr{T}$ of triples that hold in $\mathscr{M}$. $\mathscr{T}$ is defined inductively as follows, where 'dually' means 'by exchanging 0 and 1 in the definition'. For relational atoms $P(\vec{x})$, $(P(\vec{x}), X, 1) \in \mathscr{T}$ iff $P(s(\vec{x}))$ holds in $\mathscr{M}$ for every $s \in X$, and $(P(\vec{x}), X, 0) \in \mathscr{T}$ iff $P(s(\vec{x}))$ fails in $\mathscr{M}$ for every $s \in X$; and dually for $\not P$.

For the 'boolean' operators, the rules are $(\psi_0 \wedge \psi_1, X, 1) \in \mathscr{T}$ iff $(\psi_0, X, 1) \in \mathscr{T}$) and $(\psi_1, X, 1) \in \mathscr{T}$); $(\psi_0 \wedge \psi_1, X, 0) \in \mathscr{T}$ iff there are X_i such that $X = X_0 \cup X_1$ and $(\psi_i, X_i, 0) \in \mathscr{T}$ for both i; and dually for $\vee$.

For the quantifiers, $(\exists x.\psi, X, 1) \in \mathscr{T}$ iff $(\psi, X(F/x), 1) \in \mathscr{T}$ for some choice function F; and $(\exists x.\psi, X, 0) \in \mathscr{T}$ iff $(\psi, X(M/x), 0) \in \mathscr{T}$; and dually for $\forall$.

So far, the semantics looks like the natural lifting of FOL to teams, and indeed the semantics is, so far, equivalent to the FOL semantics.

The extension comes with the dependence atoms. The rule for positive dependence atoms is: $(\mathscr{D}(\vec{x}, y), X, 1) \in \mathscr{T}$ iff X makes y functional in $\vec{x}$: that is, if $s, s' \in X$ agree on $\vec{x}$, they must also agree on y; and $(\mathscr{D}(\vec{x}, y), \varnothing, 0) \in \mathscr{T}$; and dually for $\not\mathscr{D}$.

It is hard to give a good intuition for the semantics of $\not\mathscr{D}$; it is never true, except in the artifical vacuous case of the empty team (a purely technical requirement). Conversely $\mathscr{D}$ is never false (except vacuously), which one can perhaps best understand by considering that a non-functional team can always be made functional by ejecting some of its members, and it is a core property of the logic that if a team makes a formula true (or false), so does any subteam.

Note that a consequence of these rules is that the empty team makes every formula both true and false; but since it is impossible to give any intuitive meaning to a formula in the absence of any assigment, this technicality is not overly obtrusive.

On the other hand, the team consisting solely of the empty assignment, i.e. the unique non-empty team on no variables, $X = \{\varnothing\}$, plays a major role. A sentence ϕ is *true* in $\mathscr{M}$ if $(\phi, \{\varnothing\}, 1) \in \mathscr{T}$.

In Väänänen's terminology, we say X *has type* ψ iff $(\psi, X, 1) \in \mathscr{T}$.

2.3 Negation

As adumbrated in the introduction, the problem is that there are two natural understandings of negation, and though they coincide on FOL, they differ on IF. The first understanding is *game negation*: this corresponds to exchanging the roles of the players. Thus, for example, the game negation of an undetermined sentence is also undetermined, while the game negation of a true sentence is false, and vice versa. Game negation, in both the IF game semantics and the DL semantics, obeys the familiar De Morgan dualities.

The second understanding is *classical negation*: a classically negated sentence is true iff it is not the case that the unnegated sentence is true. Hence the negation of an undetermined sentence is true, thereby introducing an asymmetry.

The first careful analysis of these issues was done by Hodges [11], who also showed that classical negation could be defined in terms of game negation and an independently justifiable 'flattening' operation. In that article, Hodges used $\sim$ for game negation, and $\neg$ for classical negation. Rather unfortunately, Väänänen [16] reversed this notation. I shall

follow Hodges' notation, as it is more mnemonic ($\sim$ is smooth and symmetrical, like game negation, and $\neg$ is flat and asymmetrical, like classical negation).

As will be apparent from our description of the semantics in negation normal form, the 'standard' negation in IF and DL is the game negation. If it is included as a primitive, then the rule is $(\sim\psi, X, d) \in \mathscr{T}$ iff $(\psi, X, 1-d) \in \mathscr{T}$, and then $\wedge$ can be defined in terms of $\vee$ and $\forall$ in terms of $\exists$ by the usual dualities, and our negated atoms $\not R$ and $\not\mathscr{D}$ are just $\sim P$ and $\sim\mathscr{D}$.

We defer further discussion of classical negation until after we have introduced team-building games for the simpler case of DL.

3 Team-building games for DL

The idea is simple: although IFL and DL require either imperfect information, restrictions to uniform strategies, or games with explicit second order moves, one can consider building finite approximations to these second-order objects incrementally, essentially expressing the construction as a fixpoint operator, which we have previously studied in the context of IF; thereby, although the players build actual teams in the limit, each move in the game looks like a first-order game move.

Henceforth, we consider only **countable** models.

3.1 Game definition

As with the games for IFL, we shall define our games only for sentences. Throughout the rest of this section, let ϕ be a sentence of DL in negation normal form. Let Φ be the set of subformulas of ϕ; let *Var* be restricted to mean the variables occurring in ϕ. For a subformula ψ, let $Var(\psi)$ be the variables in scope in ψ (regardless of whether they occur in ψ). Let $\psi' \preccurlyeq \psi$ mean ψ' is a subformula of ψ.

An *annotated assignment*, or aa for short, is an assignment s together with a subformula ψ. It serves to record how the assignment was used.

A *crowd* is a set of annotated assignments, not necessarily with the same domains.

A position in the game comprises a subformula ψ, a crowd X, and an assignment s. s behaves exactly as in a first-order Hintikka game, and the crowd handles the dependency aspects.

The initial position is $(\phi, \varnothing, \varnothing)$.

The game looks like repeated plays of the first-order game, but with the addition that Eloise remembers how she played last time, and is required to play consistently with her earlier choices. The rules are:

- At position $(\psi_0 \vee \psi_1, X, s)$, Eloise checks to see whether there is $(s', \psi') \in X$ such that $\psi' \preccurlyeq \psi_i$ for some i, and $s' \restriction Var(\psi_0 \vee \psi_1) = s$. If so, there will be only one such i (which will follow from the rules), and she must choose it. Otherwise, she chooses freely $i = 0, 1$. Play moves to (ψ_i, X, s).
- For $(\psi_0 \wedge \psi_1, X, s)$, Abelard chooses freely whether to move to ψ_0 or ψ_1.
- At position $(\exists x.\psi, X, s)$, Eloise checks to see whether there is $(s', \psi') \in X$ such that $\psi' \preccurlyeq \psi$ and $s' \restriction Var(\exists x.\psi) = s$. If so, she chooses $v = s'(x)$; otherwise she has a free choice of v. Play moves to $(\psi, X, s(v/x))$.
- For $(\forall x.\psi, X, s)$, Abelard has a free choice of value for x.
- At a (negated) relational atom R ($\not R$), play stops, and Eloise wins the play iff s satisfies (fails) R.

- At a dependency atom $\psi = \mathscr{D}(\vec{x}, y)$, we check whether team built so far for ψ (recorded in the crowd X) is functional. We take the crowd $X' = X \cup \{ (s, \psi) \}$, and consider the team $Y = \{s' : (s', \psi) \in X'\}$. If Y does not satisfy ψ – i.e. y is not functional in $\vec{x}$ – then Abelard wins. Otherwise, if $(s, \psi) \in X$, then Eloise wins. Otherwise, Abelard challenges by moving to $(\phi, X', \varnothing)$.
- At a negated dependency atom $\psi = \not\mathscr{D}(\vec{x}, y)$, Abelard wins.

The intuition for the $\mathscr{D}$ rule is that during the current play, Eloise builds up a series of choices which together make a team satisfying the dependency atom. However, Abelard should have challenged her with all his possibilities; if he has exhausted his choices (in the finite case), or unnecessarily repeated something he tried earlier, he loses. (This feature is not necessary; it serves merely to force the game to be finite on a finite domain.)

The game as defined so far may, in the case of an infinite domain, not terminate, as play may pass indefinitely through dependency atoms. On such a play, Eloise wins.

This completes the definition of the game.

3.2 Remarks

The positions of the game are all finite objects, and hence the moves and finite-winning conditions are all recursive (relative to the intepretations of relational atoms), even if the domain is infinite.

As noted, if ϕ is a formula without dependence atoms (i.e. is FOL), then the game is exactly the usual Hintikka game with some additional book-keeping that is not used.

For non-FOL formulae, if the sentence ϕ is true, Eloise builds up incrementally on each play a team required to satisfy the dependency atoms, which in the DL semantics are constructed at one swoop by the quantifier semantics. In the case of a finite domain, Eloise's strategy in this game amounts to building her full strategy for the second-order DL game – on every play of this game. The consequence is that a winning strategy for Eloise in this game gives a winning strategy for DL (or for the IF game), but the same is not true for Abelard.

The game has perfect information, has simple (Büchi) winning conditions, and is therefore determined. Consequently it is clear that this game does not match the IF game. In fact, our asymmetrical treatment of the players in the rules and winning conditions amounts to making this the game simulate the second order game for the skolemized sentence, rather than the IF imperfect information game.

Because the winning conditions are Büchi, it also follows that if a player has a winning strategy, they have a history-free winning strategy, i.e. one that depends only on the current position in the game. In particular, the order in which aas are added to the crowd need not be remembered.

Since we have included a repetition detection in the winning conditions, which is not actually necessary, the game always terminates on finite domains.

3.3 Examples

3.3.1 A simple 'Snap' game

First, consider the (IF non-determined) sentence mentioned in the introduction, representing the game where Eloise and Abelard independently choose a boolean value in $M = \{0, 1\}$, and Eloise wins iff the two values are the same. In IFL, this is $\forall x.\exists y/x.x = y$; in DL, it is $\forall x.\exists y.\mathscr{D}(y) \wedge x = y$.

A sample play of the game is: A chooses 0; E chooses 0; A chooses the dependency atom, X is functional, so play continues: A chooses 1, E chooses 1, and then A will again challenge D, and E will lose because the crowd is now non-functional. This strategy of repeatedly challenging until either the accumulated crowd is non-functional or E chooses a $y \neq x$ is winning for Abelard; contrast with the IF game, where Abelard has no winning strategy.

3.3.2 An infinite game

A more interesting example arises by borrowing a well known trick for expressing the infinitude of the domain in IFL (and so incidentally demonstrating the non-FOL expressivity of IFL). In DL, the formula is

$$\exists c.\forall x.\forall u.\exists y.\exists v.\mathscr{D}(x,y) \wedge \mathscr{D}(u,v) \wedge (y = v \vee x \neq u) \wedge (x = u \vee y \neq v) \wedge (y \neq c)$$

To understand this formula, see that it asserts the existence of y that is $f(x)$, and v that is $g(u)$, and moreover $f = g$, by the first FOL clause (which says $x = u \to y = v$), and f is injective, by the second FOL clause (which says $y = v \to x = u$), and moreover f never takes the value c.

In the IFL version, the formula is true on an infinite domain, but undetermined on a finite domain of size > 2 – although Eloise can't win, she may, by blind chance, escape Abelard's attempts to detect a failure of the main clause.

Consider the team-building game in the case of an infinite domain. Eloise has a winning strategy – indeed, uncountably many winning strategies – just as in the IF game, as follows. After choosing c, she keeps in her head a suitable function f – for example, she enumerates the domain starting at c, and maps each element to the next in the enumeration. After Abelard chooses x and u, she chooses $y = f(x)$ and $v = f(u)$. Now Abelard has no chance to win in the boolean clauses, so his only hope is to challenge the dependency atoms. But whichever one he challenges, the past choices of y or v, as recorded in the crowd, are functional; so either he repeats himself and loses, or he plays for ever, and loses.

Now consider a finite domain, say $M = (0, 1, 2)$. Now Eloise cannot win; but Abelard can win, because by repeating the challenge at dependency atoms, he can force Eloise into violating either functionality, injectivity, or not hitting c.

This illustrates the theorem we shall shortly prove: Eloise wins the team-building game iff she wins the IF game, and thus iff the DL sentence is true.

3.3.3 Team-building and game negation

If our game were perfectly symmetric, we would have a contradiction (since the IF game is not determined) – how does the asymmetry we introduced solve the problem?

To see this, consider some examples involving game negation. First, consider the 'Snap' formula. If we negate the formula in DL, and push negations through to the atoms, we get $\exists x.\forall y.\not\mathscr{D}(y) \vee x \neq y$. Clearly Eloise cannot win this game: she can't win at $\not\mathscr{D}$, and Abelard will choose y to equal x. This is, indeed, a winning strategy for Abelard, although the formula is not false in IFL or DL.

Now consider the infinity example. The game negation of the formula is

$$\forall c.\exists x.\exists u.\forall y.\forall v.\not\mathscr{D}(x,y) \vee \not\mathscr{D}(u,v) \vee (y \neq v \wedge x = u) \vee (x \neq u \wedge y = v) \vee (y = c)$$

Suppose the domain is infinite. Eloise cannot win at the negated dependency atoms. If she chooses $u = x$, Abelard chooses his $y = v \neq c$ and wins; if she chooses $u \neq x$, Abelard chooses $y \neq v$ and $y \neq c$ and wins. Here we have Abelard winning a false sentence.

In the case of a finite domain, the same strategy suffices for Abelard.

To sum up, by the introduction of the asymmetry, we have arranged that Eloise wins ϕ iff ϕ is true (in the DL or IFL sense); and Eloise wins $\sim\phi$ iff ϕ is false; and if ϕ is undetermined, Abelard wins both ϕ and $\sim\phi$.

3.4 Correctness

The underlying core of the correctness theorem for the team-building game is (the dual of) Kleene's theorem that Σ^0_1-$IND = \Pi^1_1$. However, the details require some attention.

▶ **Theorem 1.** *If Eloise has a winning strategy in the team-building game for ϕ, then ϕ is DL-true, and vice versa.*

Proof. Suppose then that Eloise has a winning strategy in the team-building game for ϕ. For each subformula ψ we will construct a team $X(\psi)$ that has the type of ψ, and so that $\varnothing$ is in the team for ϕ. To do this, we will construct choice functions for Eloise, essentially giving her strategy in the second-order game for DL. The team-building game strategy does not necessarily define a unique winning strategy in the second-order game; we shall build one particular strategy.

We first make an auxiliary definition. Let X be a crowd, and ψ a formula. $X\restriction\psi$ denotes the team given as

$$\{s\restriction Var(\psi) : (s,\psi') \in X \text{and } \psi' \text{ is a subformula of } \psi\}$$

Now let T be the game tree that results from playing all Abelard's choices against Eloise's strategy. We will need to traverse this tree in a particular well-behaved order. Wlog we may assume that $M = \mathbb{N}$ (the finite case is strictly easier). A node in the game tree can be uniquely defined by a sequence of integers, giving the choices made by the players: the value of v at quantifiers, and 0 or 1 at boolean operators (although, of course, there is no Eloise branching in this tree). Order the nodes lexicographically according to this sequence (an ordering which may have length ω^ω). Call this ordering $\zeta\colon \omega^\omega \to T$.

We could define suitable teams for the first-order part directly from the semantics, so our concern is with the dependency atoms. By the rules of the game, every crowd X that appears on a dependency atom node satisfies the atom; but any such crowd has dealt with only a finite number of Abelard's possible plays. Since the union of functional teams is not necessarily functional, we need to take care when combining crowds. We will achieve this by the ordered traversal of the tree.

We will proceed by building up a crowd that contains the information required to produce teams for the DL semantics of ϕ, starting with the empty crowd.

Consider the last Abelard choice in the node labels of T (as defined above). We will explore all the possible choices, for fixed values of the earlier choices, by following an 'almost leftmost' path through T. Specifically, starting from the root, we follow a path in T by taking at each Abelard choice the leftmost unexplored choice. After making the last Abelard choice and following any subsequent Eloise choices, we are at an atomic node (ψ, X, s). If ψ is a relational atom, then s satisfies it; we backtrack to the point of the last Abelard choice, and explore the next choice. ψ cannot be a negated dependency atom, as no such nodes occur in the tree. If ψ is a dependency atom, we add s to the crowd (per the game rules), and follow on down T, repeating the earlier choices, and then exploring the next possibility for the final Abelard choice. Note that the game rules mean that Eloise cannot change her mind about her response to any of the pre-final Abelard choices, as they will be recorded in the crowd.

In the case of a finite domain, we reach a leaf, and take the crowd X_1 at that leaf. In the case of an infinite domain, this procedure may involve traversing an infinite path through T. In that case, we take X_1 to be the union of all the crowds at atomic nodes on the path. The resulting crowd does satisfy all the dependency atoms occurring on path: suppose not, then there is a dependency atom $\psi = \mathscr{D}(\vec{x}, y)$ on the path, and two assignments s, s' annotated with ψ, such that s and s' agree on $\vec{x}$ but differ on y. But since X_1 is a union of a increasing chain of crowds, there is some ψ-node on the path at which both s and s' already occur, and is therefore false, which is a contradiction.

Having generated X_1, we now restart the game with crowd X_1, and explore all the final Abelard choices for the next value of the pre-final Abelard choice. It is not immediate that Eloise's strategy can still win starting with an infinite crowd (that therefore does not appear anywhere in the actual team-building game); however, we can show that she can. Suppose not. Then Abelard has a winning play. However, all Abelard-winning plays are finite; therefore his winning play uses only a finite amount of information about the crowd, and so he can also win with this play against the finite crowd, which does appear in T.

Thus we obtain $X_2 \supseteq X_1$. We then repeat the procedure until we have exhausted the pre-final Abelard choices; then back up to consider the pre-pre-final choice for Abelard, and so on until we have exhausted all the Abelard choices. This gives a crowd X, from which Eloise can win the team-building game for ϕ; but as the crowd is exhausted, all her choices are pre-made, and the game will terminate after one round. This then gives the choices for the DL semantics: at $\exists x.\psi$, the choice function F is extracted from the crowd, and at disjunctions, the choice of disjunct is extracted from the crowd.

This completes the proof of the interesting direction.

The other direction is almost immediate: if ϕ is true, then the choice functions F at existential nodes, and the split $X = X_0 \cup X_1$ (with an arbitrary choice to make the split disjoint) give a strategy for Eloise in the team-building game. ◀

▶ **Corollary 2.** *Eloise wins the team-building game for $\sim\phi$ iff ϕ is false.*

Proof. Game negation in the DL semantics is exactly the swapping of 1 triples and 0 triples. ◀

▶ **Corollary 3.** *If ϕ is undetermined, then Abelard wins the team-building game for ϕ and also for $\sim\phi$.*

3.5 Further examples and remarks

3.5.1 The infinite game again

The infinite game of subsection 3.3.2 illustrates the proof in both directions. If we know the DL formula is true, then Eloise has choice functions defined by her 'secret' function f, and the obvious choice function at the disjunctions. On the other hand, if she is playing the team-building game, she does not even have to have the function f in her head: all she needs to do is, in each round, is to choose any y and v which will satisfy the relational part, and maintain consistency with her previous choices, thus building up the function f. Note that there is no *a priori* need for her to build the same function along different branches of the game tree; most of the work in the correctness proof was in showing that even if she doesn't, we can extract a single choice function from a modified tree.

3.5.2 Constraints on Abelard

In the original IFL, only $\exists$ quantifiers could be slashed, and only on $\forall$ variables: Abelard had no memory loss, and Eloise always remembered her own choices. Later work adopted the symmetrical approach, but many people (including myself) found it hard to hard to understand constrained Abelard choices intuitively as a logical property. The intuition that Abelard also has restricted knowledge at choices is simple enough, but one (or at least I) naturally imagines such a restriction should make it easier for Eloise to win; for, in the formal game semantics, she surely now only has to win against Abelard's uniform strategies, which should be easier than winning against all strategies.

However, this intuition is misleading: consider $\exists x.\forall y/x.\psi$. Eloise only has to win against uniform Abelard strategies – but since she doesn't know which uniform strategy Abelard is playing, that's the same as winning against any strategy. In other words, slashing Abelard variables makes it harder for Abelard to win, but not easier for Eloise to win. (As a referee put it, one needs to lose the 'truth bias' and consider truth and falsity on equal terms.) This is perhaps more easily seen in DL, thus:

In dependence logic, the dependence atoms know nothing about whether variables belong to Eloise or Abelard, and one can perfectly well write, e.g., $\exists x.\forall y.\mathscr{D}(x,y) \vee x \neq y$. This sentence is true: in the DL semantics, E's choice function at $\exists$ selects an arbitrary element v for x, and then at the $\vee$ she splits the team with (v,v) on the left and all other (v,v') on the right. In the team-building game, she chooses v (which will thus be fixed in the crowd for the rest of the play), and plays left or right at $\vee$ similarly. In this case, the game will terminate after at most two rounds, owing to our optimizing termination criterion for repeated Abelard challenges.

Here we have the alternative explanation of why Abelard slashing has been found confusing in IFL. Consider again the Snap formula: $\forall x.\exists y/x.x = y$ in IFL; $\forall x.\exists y.\mathscr{D}(y) \wedge x = y$ in DL. One naturally defines that the negation of the IF formula will be $\exists x.\forall y/x.x \neq y$. However, DL negation shows us that this means $\exists x.\forall y.\not\mathscr{D}(y) \vee x \neq y$, and since negated dependency atoms are never true, the Abelard slashing has no effect on truth, only on falsity.

3.5.3 Team-building for IFL

Since there is a simple translation from IFL (assuming that variables are not re-used) into DL, by $\exists y/\vec{x}.\psi$ mapping to $\exists y.\mathscr{D}(\mathit{Var}(\psi)\backslash\vec{x}, y) \wedge \psi$, and dually for $\forall$, the team-building game for DL immediately gives one for IFL. We can avoid the explicit translation by saying that immediately after a slashed existential quantifier, Abelard has the choice to proceed into ψ, or to start the next round. (Nothing new happens after a slashed universal quantifier, for the reasons just explained.)

4 Negation and Team Logic

4.1 Team Logic

When Väänänen defined DL, he did not use the nnf formulation that we have been using. Rather, he took game negation $\sim$ as a primitive, along with $\exists$ and $\vee$, and defined $\forall$ and $\wedge$ via De Morgan dualities with $\sim$. The semantics with fundamental triples is designed to maintain both the 'truth' and 'falsity' semantics at the same time, via the flag $d = 0, 1$.

However, when he extended DL to *Team Logic* (TL) by adding classical negation, he reverted to a traditional asymmetrical semantics, in which the denotation of a formula is a

set of teams, rather than a set of fundamental triples. Consequently, the game duals are no longer maintained automatically, and so in TL game negation does not appear explicitly, and the game dual connectives are defined as primitives, while classical $\neg$ is added as a primitive with its usual semantics.

To add to the confusion, TL changes the notation used for connectives, owing to a (slightly distorted) analogy with linear logic, and because he wishes to maintain the usual De Morgan dualities when considering *classical* rather than game negation. So the DL and IFL $\vee$ is instead written $\otimes$, and the IFL and DL $\forall$ is instead written ! (by analogy with the linear logic 'of course' operator). Then $\vee$ is reused for the classical dual of $\wedge$, and $\forall$ for the classical dual of $\exists$. (The real link between TL and linear logic is explained in [2] – it is partly linear, and partly intuitionistic.) The justification for this is perhaps that TL is really a first-order logic about teams, which involves statements about their members, whereas DL and IFL intend to be logics about members, which need teams to express certain properties.

We hope to avoid the confusion within this article by leaving our existing DL notations alone, so as not to have to reformulate the game rules, and then when required we will use subscripted versions for the new TL classical operators.

4.2 Negation in the team-building game

The team building game has the property that Eloise winning characterises truth, and Abelard winning characterises non-truth, so that at the level of entire sentences, swapping players corresponds to classical negation. However, we know from [16] that adding classical negation as a primitive to DL greatly increases its complexity, taking it to full second-order power. It is therefore interesting to explore what happens to the team-building game when we apply the usual methods of incorporating negation into a model-checking game.

Before doing so, we make one simplification to the team-building game: we drop the condition at dependence atoms that says "if $s \in X$ and X is functional, then Eloise wins". As already noted, this condition is just an optimization for the case of finite domains; its job can also be done by the "Eloise wins infinite plays" condition.

The usual way to incorporate negation into a model-checking game for a logic (e.g. in modal mu-calculus games, see e.g. [7]) is to add the De Morgan duals (w.r.t. the negation in question) of all the operators in the logic (if not there already), dualize the rules for the new operators, and dualize the winning conditions, and then deal with formulae in nnf (w.r.t. the negation in question). If we apply this methodology:

- We need the classical duals of the 'boolean' operators (where we already have both game duals in the logic). The classical dual of $\wedge$ we write as $\vee_T$ ([16] writes $\vee$); of $\vee$ we write $\wedge_T$ ([16] writes $\oplus$).
- In the semantics of TL, the DL $\forall$ and $\exists$ are self-dual under classical negation, and so no new operators are needed.
- For the relational atoms, the interpretation of $\neg$ is the same as that of $\sim$, so we need no new symbols.
- For dependency atoms, $\mathscr{D}$ under an odd number of negations is written $\bar{\mathscr{D}}$, and a $\not\mathscr{D}$ is written $\bar{\not\mathscr{D}}$.

The rules for the operators are dualized by exchanging 'Eloise' and 'Abelard'.

Dualizing the winning conditions raises a number of obstacles in the classically negated dependencies $\bar{\mathscr{D}}$. The natural dualized rule would have Eloise winning as soon as a functional crowd (i.e. a crowd not satisfying the atom) is constructed. This, of course, is wrong – the first time through the atom, there will be one aa in the crowd, which is necessarily functional.

Instead, we need to allow Eloise to keep trying to construct a non-functional team; if she fails forever, then Abelard will win. So the rule for $\bar{\mathscr{D}}$ simply moves the game to the start of the next round.

The real problem comes from the infinite plays. The team-building game has Eloise winning on infinite plays involving $\mathscr{D}$. We did not need to worry about which particular $\mathscr{D}$ occurs infinitely often, because if Abelard can break any allegedly functional Eloise team, he can do it in finite time, and his best strategy is to do so. However, if Abelard needs to win infinite plays involving $\bar{\mathscr{D}}$, we are faced with the problem of plays involving both infinitely many $\mathscr{D}$s and infinitely many $\bar{\mathscr{D}}$s. While one might think that a sufficiently clever intertwining might solve the problem, it is in fact possible to prove that there is no (reasonable) solution to this problem.

▶ **Theorem 4.** *Given an extended team-building game as indicated, then there is no first-order, or even uniform second-order, way to define winning infinite plays such that the game captures Team Logic.*

Proof. It is a standard theorem (see, e.g. [12]) that if the winning conditions of a (standard) perfect information game are Σ^1_n, then the winning sets for the game are at most Π^1_{n+1}. Since TL expresses all second-order properties, no second-order winning condition of fixed quantifier depth suffices. ◀

However, while it is impossible to reach SOL with standard games, it may be possible with richer games (other than by going back to imperfect information). We

▶ Conjecture 5. The team-building game can be extended to a game with transfinitely (but still countably) long plays, with first-order winning conditions, so as to capture Team Logic.

5 Conclusion

In this article, we have shown how it is possible to design a game semantics for DL (and hence IFL) that preserves the intuition of first-order games, but does not resort to imperfect information. The game is thus determined, and so characterizes truth and non-truth, rather than truth and falsehood.

We discussed why the natural attempts to extend such games to full Team Logic must fail, and conjectured that nonetheless they can be extended with the (already studied) idea of transfinite plays.

Future work is to investigate the conjecture further, and also explore the use of similar team-building games in the logic IF-LFP (independence logic with fixpoints), a logic which is also second-order expressive, and has not received any game characterization previously.

Acknowledgements. This paper grew out of the Dagstuhl Seminar 13071 'Dependence Logic: Theory and Applications'.

I thank an anonymous referee for helpful corrections and suggestions.

References

1 S. Abramsky, J. Kontinen, J Väänanen, H. Vollmer, Dependence Logic: Theory and Applications (Dagstuhl Seminar 13071), *Dagstuhl Reports*, **3**(2) 45–54, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, `http://dx.doi.org/10.4230/DagRep.3.2.45` (2013).

2 S. Abramsky, J. A. Väänanen, From IF to BI: a tale of dependence and separation. CoRR abs/1102.1388 (2011)

3 J. C. Bradfield, The modal mu-calculus alternation hierarchy is strict, *Theor. Comput. Sci.* **195** 133–153 (1997).

4 J. C. Bradfield, Independence: logics and concurrency. In: Tuomo Aho and Ahti-Veikko Pietarinen (eds). Truth and Games: Essays in Honour of Gabriel Sandu (Acta Philosophica Fennica 78) 47–70. Helsinki: Societas Philosophica Fennica. (2006)

5 J. C. Bradfield and S. Kreutzer, The complexity of independence-friendly fixpoint logic, in: Stefan Bold, Benedikt Löwe, Thoralf Räsch, Johan van Benthem (eds.), Foundations of the Formal Sciences V, Infinite Games 39-62. [Studies in Logic 11]. London: College Publications (2007).

6 J. C. Bradfield and S. B. Fröschle, Independence-friendly modal logic and true concurrency, *Nordic J. Computing* **9** 102–117 (2002).

7 J. C. Bradfield and C. Stirling, Modal Mu-Calculi, in P. Blackburn, J. van Benthem and F. Wolter, *Handbook of Modal Logic*,721–756, Elsevier (2007).

8 H. B. Enderton, Finite partially ordered quantifiers, *Z. für Math. Logik u. Grundl. Math.* **16** 393–397 (1970).

9 L. Henkin, Some remarks on infinitely long formulas, *Infinitistic Methods*, Pergamon Press, Oxford and PAN, Warsaw, 167–183 (1961).

10 J. Hintikka and G. Sandu, A revolution in logic?, *Nordic J. Philos. Logic* **1**(2) 169–183 (1996).

11 W. Hodges, Compositional semantics for a language of imperfect information, *Int. J. IGPL* **5**(4), 539–563.

12 Y. N. Moschovakis, *Descriptive set theory*. North-Holland, Amsterdam & New York (1980).

13 A. L. Mann, G. Sandu and M. Sevenster, Independence-Friendly Logic – A Game Theoretic Approach, Cambridge University Press (2011).

14 Marc Pauly, Logic for Social Software. Ph.D. Thesis, Universiteit van Amsterdam (2001).

15 G. Sandu, On the logic of information independence and its applications, *J. Philos. Logic* **22** 361-372 (1993).

16 J. Väänänen, *Dependence Logic*. Cambridge University Press (2007).

17 W. J. Walkoe, Jr, Finite partially-ordered quantification. *J. Symbolic Logic* **35** 535–555 (1970).