Contextual Intensional Logic: Type-Theoretic and Dynamic Considerations

Richmond H. Thomason
Philosophy Department
University of Michigan
Ann Arbor, MI 48109-2110, USA

Email: rich@thomason.org

June 27, 2001

Abstract

This paper presents Contextual Intensional Logic, a type-theoretic logic intended as a general foundation for reasoning about context. I motivate and illustrate the logical framework, and discuss a dynamic version that provides for context-changing operators.

1. Introduction

In several previous works, [Thomason, 1997, Thomason, 1998], I proposed and explored (very briefly) the idea of formulating contextual logic as a version of type theory. This paper will improve, refine and extend these type-theoretic ideas.

There are a number of logical advantages to type theory.

(1) The underlying logical architecture, which goes back to [Church, 1940], is beautifully simple and has been thoroughly investigated by logicians.

(2) The framework of types provides a rich, highly structured ontology that is potentially useful in formalization.

(3) The theory is a straightforward extension of Richard Montague's Intensional Logic (IL), [Montague, 1970, Gallin, 1975, Anderson, 1984], which has been the dominant formalism for the logical interpretation of natural language. Using it provides direct connections to an extensive body of work in natural language semantics. So the type theoretic approach can facilitate linguistic applications of the theory of context.

(4) The use of types provides conceptual clarity.

In this paper, I will introduce and explain the basic ideas of the formalism, using these to illustrate and support point (4), above. I will also discuss some ways in which the formalism can (and should) be extended, with more detailed remarks concerning dynamic extensions of the logic than those that appeared in [Thomason, 1999], the conference publication of this paper.

2. Brief Introduction to IL

Any version of type theory will involve—in addition to a domain of individuals, with variables ranging over this domain—domains that correspond to higher-order types: sets of individuals, sets of sets of individuals, etc. The formalizations of type theory based on [Church, 1940] use functional (i.e., lambda) abstraction to organize the higher-order domains. In general, where D_1 and D_2 are domains, the set $D_2^{D_1}$ of functions from D_1 to D_2 is also a domain.

This leads to the following recursive definition of types, in which the types for individuals and truth values are primitive, and all other types are functional.

(2.1) e is a type.
(2.2) t is a type.
(2.3) If σ and τ are types, so is $\langle\sigma,\tau\rangle$.

Here, e is the type of individuals (entities), t is the type of truth-values, and $\langle\sigma,\tau\rangle$ stands for the type of functions from objects of type σ to objects of type τ.

The language of type theory has an infinite set of variables of each type.[1] The language has only three primitive syntactic constructions.

Identity:	If ζ and ξ are expressions of type τ, so is $\zeta=\xi$.
Functional application:	If ζ is an expression of type $\langle\sigma,\tau\rangle$ and ξ is an expression of type σ, then $\zeta(\xi)$ is an expression of type τ.
Lambda abstraction:	If ζ is an expression of type τ, then $\lambda x_\sigma\zeta$ is an expression of type $\langle\sigma,\tau\rangle$.

With these resources, the full set of boolean operations can be defined, as well as universal and existential quantification over domains of any type. The model theory of the logic is straightforward; arbitrary domains are assigned to primitive types, the domain of $\langle\sigma,\tau\rangle$ is the set of functions from the domain of type σ to the domain of type τ, $=$ is interpreted as identity, $()$ is interpreted as functional application, λ is interpreted as functional abstraction. See [Gallin, 1975] for details on these matters.

A number of ontological policies come along with this approach to types: sets are represented by the corresponding characteristic functions. That is, a set of objects of type τ is represented as a function of type $\langle\tau,t\rangle$, and n-place functions from types $\langle\sigma_1,\sigma_2,\ldots,\sigma_n\rangle$ to

[1]In writing formulas of intensional logic, I will label the first occurrence of a variable with its type unless the type is e, in which case the label may be omitted. Later occurrences will not be marked for type; no confusion can arise, as long as all independent uses of bound variables involve distinct variables.

type τ are represented as a nested type $\langle\sigma_1, \langle\sigma_2, \ldots \langle\sigma_n, \tau\rangle \ldots\rangle\rangle$ consisting of iterations of 1-place functions. According to these policies, for instance, a 2-place function from individuals to individuals would have type $\langle e, \langle e, e\rangle\rangle$; a function of this type inputs an individual and outputs a function from individuals to individuals. A 2-place relation between individuals and sets of individuals would have type $\langle e, \langle\langle e, t\rangle, t\rangle\rangle$. An object of this type would be a function that inputs a first-order object and outputs a function that inputs a set of first-order objects (which itself is a function from first-order objects to truth-values) and outputs a truth-value.

Church's formalism uses the domain of truth values to interpret sentences. Obviously, a domain containing only two values will be unable to represent sentence meanings adequately, an inadequacy that is reflected in the inability of the logic to deal with *propositional attitudes* such as belief. Montague's IL seeks to remedy this problem by introducing a third primitive type w, the type of possible worlds. This makes available a type $\tau\text{-Prop} = \langle w, t\rangle$ of propositions.[2]

In natural language, we apparently do not refer explicitly to possible worlds; similarly, the syntax of IL gives worlds an implicit role. There are no constants or variables of type w, so there is no explicit lambda abstraction over the domain of worlds. Instead, IL has operators that form intensions and extensions.

Intension:	If ζ is an expression of type τ, then ${}^\wedge\zeta$ is an expression of type $\langle w, \tau\rangle$.
Extension:	If ζ is an expression of type $\langle w, \tau\rangle$, then ${}^\vee\zeta$ is an expression of type τ.

It is time to be more explicit about the model theory. A *frame* $\mathcal{F}$ assigns a nonempty set $\text{Dom}_{\mathcal{F}}(\tau)$ to each primitive type $\tau \in \{e, w\}$; $\text{Dom}_{\mathcal{F}}(t) = \{\top, \bot\}$, where $\top$ and $\bot$ are two arbitrary, fixed objects. The domains assigned to complex types in $\mathcal{F}$ are then defined by the following equation.

$$\text{Dom}_{\mathcal{F}}(\langle\sigma, \tau\rangle) = \text{Dom}_{\mathcal{F}}(\tau)^{\text{Dom}_{\mathcal{F}}(\sigma)}.$$

A model $\mathcal{M}$ on $\mathcal{F}$ assigns each constant α of type τ a value $[\![\alpha]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}}$ in $\text{Dom}_{\mathcal{M}}(\tau) = \text{Dom}_{\mathcal{F}}(\tau)$, for each world $\mathrm{w} \in \text{Dom}_{\mathcal{M}}(w)$ and index $\mathrm{i} \in \text{Dom}_{\mathcal{M}}(i)$; $\mathcal{M}$ assigns each variable x of type τ a value $[\![x]\!]_{\mathcal{M}}$ in $\text{Dom}_{\mathcal{M}}(\tau)$; for all w and i, $[\![x]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}} = [\![x]\!]_{\mathcal{M}}$.

The values of complex expressions are defined as follows, where $\mathcal{M}^{\mathrm{d}}/x$ is the model like $\mathcal{M}$ except that $[\![x]\!]_{\mathcal{M}^{\mathrm{d}}/x} = \mathrm{d}$.

Where ζ and ξ are expressions of type τ, $[\![\zeta = \xi]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}} = \top$ iff $[\![\zeta]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}} = [\![\xi]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}}$.

Where ζ is an expression of type $\langle\sigma, \tau\rangle$ and ξ is an expression of type σ, $[\![\zeta(\xi)]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}} = [\![\zeta]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}}([\![\xi]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}})$.

Where ζ is an expression of type τ, $[\![\lambda x_\sigma(\zeta)]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}} = \mathrm{f}$, where f is the function from $\text{Dom}_{\mathcal{M}}(\sigma)$ to $\text{Dom}_{\mathcal{M}}(\tau)$ such that $\mathrm{f}(\mathrm{d}) = [\![\zeta]\!]_{\mathcal{M}^{\mathrm{d}}/x,\mathrm{i},\mathrm{w}}$.

[2]Actually, Montague's type apparatus extends the type-forming operators rather than the primitive types. But it is equivalent to the version I present here; see [Gallin, 1975] for details.

Where ζ is an expression of type τ, $[\![{}^{\wedge}\zeta]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}} = \mathrm{f}$, where f is the function from $\mathrm{Dom}_{\mathcal{M}}(w)$ to $\mathrm{Dom}_{\mathcal{M}}(\tau)$ such that $\mathrm{f}(\mathrm{w}') = [\![\zeta]\!]_{M,\mathrm{w}',\mathrm{i}}$.

Where ζ is an expression of type $\langle w, \tau\rangle$, $[\![{}^{\vee}\zeta]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}} = [\![\zeta]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}}(\mathrm{w})$.

Treating propositions as sets of possible worlds is, of course, problematic. (See, for instance, [Moses, 1988, Giunchiglia *et al.*, 1993].) But despite its foundational problems, this approach has been pursued with some success in philosophical logic, computer science, economics, and natural language semantics. It is certainly possible to generalize the possible worlds approach to intensionality to obtain a less restrictive account of propositions. But it seems to me that such generalizations are premature. Without constraints, hyperintensional theories are uninformative. Appropriate constraints seem to require better general models of the reasoning agent than we have at present. Until such models are developed, I prefer to use the possible worlds formalisms, which in any case have many features that would need to be preserved in any more general approach.

Richard Montague showed that the framework of intensional logic provides a type for propositions, and that many other types that are ontologically natural and useful in the interpretation of natural language semantics can be characterized in IL. I will now explain why this framework also provides an appropriate and useful architecture for thinking about contexts.

3. Contexts as Modalities

The integration of knowledge sources and modules seems at present to be the origin of the most detailed and illuminating examples of ways in which the theory of context can be used.[3] If we look at a context as a knowledge source, the simplest way to model contexts type theoretically would be to identify a context with the set of propositions that it delivers. This would locate contexts in the type $\langle\langle w, t\rangle, t\rangle$, which is the type that I assigned to contexts in [Thomason, 1998].

This type-theoretic account of contexts would suffice for applications in which an agent (which is not itself a context) is simultaneously accessing information from many contexts. Each context would then deliver its set of propositions to the collecting agent. But if we want to allow *contexts* to access information from other contexts, we need an enriched representation of contexts.

The problem is this. The type $\langle\langle w, t\rangle, t\rangle$ makes contexts have propositions as inputs. This intensionality of contexts is mandated by the desired applications: we certainly do not want a context to support *every* true sentence if it supports *any* true sentence. Since the output assigned to a proposition by a context c is only a truth value according to this account, although we can access a proposition p that holds in c there will be no way to construct from c and p the proposition that says that p holds in c. Therefore, we can't pass the output of applying a context to a proposition on to another context as an input.

To solve this problem, we need to enrich the output of the type that is associated with contexts. Iteration of contexts can be managed in several ways within intensional logic; the following formalization corresponds directly to multi-agent epistemic logics, of the sort

[3]See, especially, the example in [McCarthy and Buvač, 1995, Section 6].

discussed in [Fagin *et al.*, 1995], in which a binary relation $\mathrm{R_a}$ over worlds is associated with an epistemic agent a.

This standard relational semantics for modal operators specifies that $\Box_a\phi$ is true in w if and only if ϕ is true in all worlds w′ such that $\mathrm{wR_aw'}$. The idea can be captured in IL by (i) locating contexts in the type $\tau_1\text{-Con} = \langle\langle w,t\rangle,\langle w,t\rangle\rangle = \langle\tau\text{-Prop},\tau\text{-Prop}\rangle$ and (ii) for each context M adding following axiom, which guarantees that the behavior of the context is determined according to the standard constraint on modalities.

$$(3.1)\ \ \forall p_{\tau\text{-Prop}}[{}^{\vee}[M({}^{\wedge}[{}^{\vee}p \to {}^{\vee}q])] \to [{}^{\vee}M(p) \to {}^{\vee}M(q)]]$$

Here, M is a constant of type τ-Mod.

Within this framework we can provide a definition of McCarthy's *ist* relation, reconstructed here as a relation between contexts and the *propositions* (not the sentences) that hold in these contexts.

$$(3.2)\ \ ist_1 \ = \ \lambda x_{\tau\text{-Mod}}\lambda y_{\tau\text{-Prop}}\, x(y)]$$

This definition gives *ist* the type $\langle\tau\text{-Mod},\langle\tau\text{-Prop},\tau\text{-Prop}\rangle\rangle$ which inputs a modality and a proposition, and outputs a proposition.

This approach to context has several shortcomings, some of them substantive and some of them a matter of public relations. I will address the less important class of problems first, leaving the substantive issues for the remaining sections.

If we treat contexts as modal operators, it is hard to see what is new about the logic of context. Modal logic is a well developed area of logic that has received a great deal of attention over the last forty years. Regarding contextual logic as a branch of modal logic seems to leave relatively little work for us to do.

While it is true that this conservative approach may rule out a more logically creative program, it still leaves room for some innovations, because (as I will argue in the next section), the logic of context can't in fact be identified with modal logic. But treating it as a generalization which preserves the main features of modal logic has many advantages. First, it enables us to import results and applications from modal logic. It is in fact, very useful to regard contexts as simple epistemic agents, agents which know information about other agents and can communicate with other, similar agents. This makes ideas concerning protocol design and knowledge-based programming available in the logic of context.[4] We may also be able to import the techniques that have been developed for modal theorem proving. (See [Stone, 1998].)

4. Contextual Intensional Logic

There is a serious limitation to this approach to context; it will not deal with cases in which the meanings of terms can differ from context to context. The computational literature has regarded variation in meaning as an essential application of the logic of context. But modal logic can't track shifts of meaning among the various senses of ambiguous expressions. Take the simplest case: two "personal assistant" databases with two users, a and b. Both

[4]See [Fagin *et al.*, 1995] for discussion of these matters, and for further references.

databases record information about their users, using an internal constant @USER to refer to their users. To merge this information coherently, we have to assign different propositions to expressions like

$$\text{(4.1)}\quad \textsc{Birthdate}(@\textsc{user}, \langle 4, 4, 1969\rangle).$$

But modal logic has no natural way to represent the reasoning that produces these different assignments.[5]

Translating ideas from David Kaplan [Kaplan, 1978] into the type-theoretic framework, I propose to address this problem by introducing a fourth primitive type: the type i of indices. Kaplan thinks of indices as arguments that determine the interpretations of "indexical expressions" such as 'I', 'here', and 'now'.

Indexicals are certainly a dramatic illustration of context-dependence, but typical uses of contextual reasoning to integrate knowledge sources also need to be able to deal with the ambiguity of lexical expressions that are not usually thought of as indexical; for instance, a case in which one database used by a bank uses 'Account' to refer to active banking accounts, while another database used by the same bank uses 'Account' to refer to active and inactive banking accounts, while still a third database uses 'Account' to refer to brokerage accounts.

I want to extend the range of Kaplan's indices, by thinking of an index as a simultaneous disambiguation of all the relevant lexical indexicalities and ambiguities that can arise in an application. Call such a simultaneous disambiguation a *contextualization*. If the only contextualizations arise from 'I' and 'here', we can identify an index with a pair consisting of a person and a place. If the only contextualizations serve to resolve the ambiguity of ten lexical items, each of them having just two possible meanings, then we will need indices corresponding to the 2^{10} possible disambiguations.

As a first step in presenting Contextual Intensional Logic (CIL), we extend the types by introducing a primitive type i for indices. The recursive definition of complex types then provides for types such as $\langle i, \langle w, t\rangle\rangle$ (the type of a context-dependent proposition) and $\langle i, \langle w, \langle e, t\rangle\rangle\rangle$ (the type of a context-dependent property).

We will provide operators $^\sqcap$ and $^\sqcup$, analogous to the corresponding operators used to manage intensionality in IL. The interpretation of expressions involving these operators is as follows.

Where ζ is an expression of type τ, $[\![{}^\sqcap\zeta]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}} = \mathrm{f}$, where f is the function from $\mathrm{Dom}_{\mathcal{M}}(i)$ to $\mathrm{Dom}_{\mathcal{M}}(\tau)$ such that $\mathrm{f}(\mathrm{i}') = [\![\zeta]\!]_{\mathcal{M},\mathrm{i}',\mathrm{w}}$.

Where ζ is an expression of type $\langle i, \tau\rangle$, $[\![{}^\sqcup\zeta]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}} = [\![\zeta]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}}(\mathrm{i})$.

The treatment of the type indices in CIL, however, is not entirely similar to that of worlds in Montague's IL. We will allow constants and variables of type i, and explicit lambda abstraction over these types. This is needed in order to provide a means of defining functions that return context-dependent values. Definition (4.3), below, of McCarthy's *ist* construction, is an example.

[5] We could, of course, formalize the reasoning as syntactic, i.e. we could treat it as reasoning about expressions. I will not explore this alternative here; despite its apparent naturalness, it is much less satisfactory in the long run, I believe, than the intensional approach that I assume here. The main formal problem that a syntactic approach raises, of course, is that it reintroduces the semantic paradoxes.

Following Kaplan, I want to think of the evaluation of contextualized expressions as taking place in two phases: (1) a disambiguation phase, where the expression is assigned an intension and (2) an evaluation phase, where the intension is evaluated in a world. Thus, to interpret an expression like

(4.2) 'I'm over 21 years old'

we first need to identify the speaker, s; this yields the proposition that is true in a world if and only if s is over 21 years old in that world. We can think of this as the two-stage evaluation of an abstract representation of the sentence's meaning called its *character*, a representation which captures the potentiality of a sentence meaning to yield different propositions in different "contexts" by treating the character as a function from indices to propositions. In general, the character is first evaluated at an index to yield a *content*, which then may be evaluated in a world to produce an extension. In CIL, it is natural to assign sentences a character of type $\tau\text{-Char-Prop} = \langle i, \tau\text{-Prop}\rangle$; this type will input an index and output a proposition.

Now, a context of the sort envisaged by McCarthy will perform two different functions: it will serve as a source of disambiguation and as a local knowledge source. From the standpoint of CIL, the first of these functions is represented by an index, or object of type i; the second, as before, is represented by a modality, or object of type $\tau\text{-Mod}$. *Therefore, a context is a pair consisting of an index and a modality.*

This representation yields no very natural type for contexts in CIL, because CIL provides no clean encoding for ordered pairs of objects of different type. However, if we use one of the (unnatural) encodings in CIL of the cross product $\sigma \times \tau$ of σ with τ, or (better) if we add a cross-product operation to the underlying type definition, we can situate *indexical contexts* in the type $\tau_2\text{-Con} = i \times \tau_1\text{-Con}$.

But we can avoid the need to provide a type for indexical contexts by separating indices and modalities as arguments of *ist*, making it, say, a function from objects of type i to functions from objects of type $\tau\text{-Mod}$ to appropriate values. If we want this *ist* to iterate, it should output a propositional character. The most general account of *ist*, then, would give it the type $\langle i, \langle \tau\text{-Mod}, \langle \tau\text{-Char-Prop}, \tau\text{-Char-Prop}\rangle\rangle\rangle$. This *ist* inputs a modality M and an index i. Then, given a propositional character p, it outputs a propositional character q, which at any index j expresses the proposition that p expresses a proposition at i that is necessary according to M. Therefore, expressions of the form $ist(\eta)(\zeta)(\phi)$ designate constant characters, ones that express the same proposition at each index. So we can in fact assign the simpler type

$$\langle i, \langle \tau\text{-Mod}, \langle \tau\text{-Char-Prop}, \tau\text{-Prop}\rangle\rangle\rangle$$

to *ist*. Accordingly, *ist* outputs a proposition rather than a propositional character. To iterate this *ist*, we use the $^\sqcap$ operator, as follows:

$$ist(\eta)(\zeta)(^\sqcap ist(\eta')(\zeta')(\phi)).$$

This simplified *ist* can be defined as follows using lambda abstraction.

(4.3) $ist = \lambda u_i \lambda x_{\tau\text{-Mod}} \lambda p_{\tau\text{-Char-Prop}}\, x(p(u))$

According to this definition, given an index i and a modality M, *ist* inputs a propositional character p and outputs the proposition that the proposition expressed by p at i is necessary according to M.

These ideas are contrary in spirit to remarks in which McCarthy suggests that contexts should be formally treated as primitives. I take these remarks to mean that, although in some applications we can axiomatize general knowledge about contexts, it is pointless to attempt to define contexts. Actually, I agree with McCarthy that relatively little of the work that needs to be done to explicate contexts can be done with definitions. But I do think it is enlightening and helpful to separate contexts into two components, one of them (the index) dealing with indexicality and ambiguity, while the other (the modality) deals with knowledge.

To show how the intended sort of applications would be formalized in CIL, it will be helpful to present a simple example. A group of people including Ann, Bob and Charlie use personal databases to manage their calendars. The language of the databases has a first-person pronoun referring to the database user, as well as constants referring to Ann, Bob and Charlie. In Ann's database, 'I' refers to Ann. Assume that the databases contain information about meetings, in the form of a set of triples whose whose first and second members denote people, and whose third member denotes a time. They also contain information about databases, in the form of a set of quadruples whose first member denotes a database, whose second and third members are people, and whose fourth member denotes a time. In this example, we do not distinguish between people and their databases; a constant a of type e refers to Ann (and to her database); similarly, b refers to Bob and c to Charlie.

Ann's database contains the following items.

(ADB1.1) $\langle \mathrm{I}, b, 9\rangle$
(ADB1.2) $\langle \mathrm{I}, c, 10\rangle$
(ADB1.3) $\langle b, \mathrm{I}, b, 9\rangle$

Without using the contextual apparatus of CIL, the knowledge state of the database can be represented using the following axioms. Here, $[\,a\,]$ and $[\,b\,]$ are constants of type τ-Mod, representing the modalities corresponding to Ann's and Bob's databases.[6]

(ADB2.1) $[\,a\,]\textsc{Meet}(\mathrm{Ann}, \mathrm{Charlie}, 9)$
(ADB2.2) $[\,a\,]\textsc{Meet}(\mathrm{Ann}, c, 10)$
(ADB2.3) $[\,a\,][\,b\,]\textsc{Meet}(\mathrm{Ann}, b, 9)$

These axioms correspond to a natural way of talking about what Ann (or her database) knows: Ann knows that she is to meet with Bob at 9, Ann knows that she is to meet with Charlie at 10, Ann knows that Bob knows she is to meet with Bob at 9. However, such a formalization involves translating Ann's indexical representation into neutral terms. In this simple example that is not a difficult problem, but it can be a challenge in more complicated cases. Moreover, it is plausible to assume that (ADB2.3) holds because Ann's and Bob's database are communicating. When this happens, Ann's database somehow has to impose

[6] $\textsc{Meet}$ has type $\langle e, \langle e, \langle e, t\rangle\rangle\rangle$. $\textsc{Meet}(\eta, \zeta, \xi)$ is an abbreviation of $[[\textsc{Meet}(\eta)](\zeta)](\xi)$.

the right interpretation on the presence of $\langle \mathrm{I}, a, 9\rangle$ in Bob's database; somehow, Ann has to know that 'I' refers to Bob when Bob uses it.

These are problems of the sort that McCarthy's *ist* is designed to solve. The somewhat different account of *ist* in CIL performs a similar function. First, in this example we identify indices with people; $\mathrm{Dom}_{\mathcal{M}}(i) = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$, where $\mathrm{a}, \mathrm{b}, \mathrm{c} \in \mathrm{Dom}_{\mathcal{M}}(e)$. Now, nothing prevents domains of different types from overlapping, but each expression must have a unique type; so we need different constants for people considered as individuals and for people considered as indices. We will use a_e, b_e, and c_e for the former, and a_i, b_i, and c_i for the latter. Although a_e and a_i, for instance, both denote the same thing, $a_e = a_i$ is not a well-formed expression of CIL.

We then let I be a constant of type e such that $[\![I]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}} = \mathrm{i}$ for all $\mathrm{i} \in \mathrm{Dom}_{\mathcal{M}}(i)$; ${}^{\ulcorner}I$ returns a in Ann's database, b in Bob's database, and c in Charlie's database; in other words, I is assigned the character intuitively associated with the first person singular pronoun.

Using *ist*, we reaxiomatize Ann's database as follows.

(ADB3.1) ${}^{\vee}ist(a_i, [\,a\,], {}^{\ulcorner\wedge}\textsc{Meet}(I, b_e, 9))$[7]
(ADB3.2) ${}^{\vee}ist(a_i, [\,a\,], {}^{\ulcorner\wedge}\textsc{Meet}(I, c_e, 10))$
(ADB3.3) ${}^{\vee}ist(a_i, [\,a\,], {}^{\ulcorner}[ist(b_i, [\,b\,], {}^{\ulcorner\wedge}\textsc{Meet}(I, b_e, a_e))])$

Recall that in this example, we are identifying indices with people; and we can use an appropriate modality to represent what each of the two databases knows. Let $[\,a\,]$ and $[\,b\,]$ express the modalities corresponding to Ann's and Bob's databases; also recall that these modalities are functions from propositions to propositions. Then $ist(a_i, [\,b\,], {}^{\ulcorner\wedge}\textsc{Meet}(b_e, \mathrm{I}, 9))$, for instance, says that the propositional character expressed by 'Bob has a meeting with me at 9' expresses a proposition in Bob's database that this database knows.

The reaxiomatization using *ist* permits a formalization of how (ADB2.3) can be inferred from observing the syntactic information that is available in Bob's database. The steps in this reasoning require the following axioms.

(DB.1) $\Box({}^{\wedge}\forall x_e \forall y_e[[{}^{\ulcorner\wedge}\textsc{Meet}(I, x, y)](b_i) = {}^{\wedge}\textsc{Meet}(b_e, x, y)])$
(DB.2) $\Box({}^{\wedge}\forall x_e \forall y_e[\,[{}^{\ulcorner}[\,[\,b\,]({}^{\wedge}\textsc{Meet}(I, x, y))]](a_i) = [{}^{\ulcorner}[\,[\,b\,]({}^{\wedge}\textsc{Meet}(a_e, x, y))]](a_i)])$
(DB.3) $\Box({}^{\wedge}\forall x_e \forall y_e \forall z_e[\textsc{Meet}(x, y, z) = \textsc{Meet}(y, x, z)])$

Axiom (DB.1) is one of several schemas that constrain the characters of literals involving the first person pronoun; in this case, it implies that what 'I have a meeting with Ann at 9' expresses for Bob is the same proposition as that expressed (for any agent) by 'Bob has a meeting with Ann at 9'. The necessity sign in (DB.1) is universal necessity, so (DB.1) implies that all databases know this equivalence. Axiom (DB.2) says that for Ann, 'I' means Ann even in statements about what Bob knows.[8] Schemas of this sort are analogous to McCarthy's "lifting axioms;" they allow the reformulation of index-dependent information from one index to another. Finally, Axiom (DB.3) states the symmetry of $\textsc{Meet}$ in the first and second arguments.

The formalization of a train of reasoning that could lead Ann to conclude that Bob knows he has a meeting with her at 9 from observing Bob's database proceeds as follows.

[7] $ist(\eta, \zeta, \xi)$ abbreviates $ist(\eta)(\zeta)(\xi)$.

[8] This last axiom is somewhat overspecialized. I believe that it can be made to follow from more general principles, but will not try to do that here.

(i) Ann's database observes the triple $\langle \mathrm{I}, a, 9\rangle$ in Bob's database.

(ii) The knowledge that Ann's database obtains from this observation is readily formalized as

$$^{\vee}ist(a_i, \mathtt{[}\,a\,\mathtt{]}, {}^{\ulcorner}ist(b_i, \mathtt{[}\,b\,\mathtt{]}, {}^{\ulcorner\wedge}\textsc{Meet}(I, a_e, 9))).$$

Note that there is a simple, straightforward syntactic mapping between the terms of the observed triple and those of the formalization. This relationship is made possible by the use of characters.

(iii) Using the definition of *ist*, we obtain

$$^{\vee}ist(a_i, \mathtt{[}\,a\,\mathtt{]}, {}^{\ulcorner}\mathtt{[}\,b\,\mathtt{]}([{}^{\ulcorner\wedge}\textsc{Meet}(I, a_e, 9)](b_e)))$$

from (ii) by lambda abstraction.

(iv) From (iii), we obtain

$$^{\vee}ist(a_i, \mathtt{[}\,a\,\mathtt{]}, {}^{\ulcorner}[\mathtt{[}\,b\,\mathtt{]}({}^{\wedge}\textsc{Meet}(b_e, a_e, 9))])$$

using (DB.1).

(v) (DB.3) then yields

$$^{\vee}ist(a_i, \mathtt{[}\,a\,\mathtt{]}, {}^{\ulcorner}[\mathtt{[}\,b\,\mathtt{]}({}^{\wedge}\textsc{Meet}(a_e, b_e, 9))]).$$

(vi) From (v), using (DB.2) and the definition of *ist*, we obtain

$$^{\vee}ist(a_i, \mathtt{[}\,a\,\mathtt{]}, {}^{\ulcorner}[\mathtt{[}\,b\,\mathtt{]}({}^{\wedge}\textsc{Meet}(I, b_e, 9))]).$$

(vii) Finally, (vi) can be used to justify the occurrence of $\langle b, \mathrm{I}, b, 9\rangle$ in Ann's database.

I hope that this example, simple as it is, gives a sense of how CIL could be useful in specifying inter-contextual reasoning.

5. Getting Dynamic

In [McCarthy and Buvač, 1995, McCarthy and Buvač, 1998], McCarthy and Buvač suggest that operations of entering and exiting a named context would be useful additions to the logic of context, adding the thoughts that these operations should be analogous to pushing and popping a stack, and that there should be analogies between contexts and the subproofs of the Fitch-style natural deduction format ([Fitch, 1952]). Similar remarks can be found in [McCarthy, 1993]. The presentations that I have seen of these ideas are intriguing, but somewhat sketchy.

Ideas from dynamic logic (see, for instance, [Harel, 1984]) provide natural and principled methods for extending a purely declarative logic like CIL to one that contains dynamic operators. One advantage of this approach is that it provides a model theoretic interpretation of the resulting semantics.

In this section, I will indicate in a preliminary way how these techniques can be applied to CIL to obtain a version of Dynamic Contextual Intensional Logic (DCIL) with operators of the sort envisaged by McCarthy and Buvač.

A dynamic version of Intensional Logic is presented in [Groenendijk and Stokhof, 1990]. This project was motivated by the desire to account for anaphora in natural language using dynamic variable binding. The requirements of dynamic contextual reasoning are rather different, and the logic DCIL presented below does not share many points of similarity with Groenendijk and Stokhof's Dynamic Montague Grammar.

Recall that in CIL an expression ζ is assigned a value $[\![\zeta]\!]_{\mathcal{M},\mathrm{i},\mathrm{w}}$ in a model $\mathcal{M}$; this value depends on a world w and an index i. We now revise this interpretation by adding a modality M in $\mathrm{Dom}_{\mathcal{M}}(\tau\text{-Mod})$ as another parameter, so that ζ is now assigned $[\![\zeta]\!]_{\mathcal{M},\mathrm{M},\mathrm{i},\mathrm{w}}$. Constants and complex expressions are interpreted as before, so the interpretation of expressions does not depend on the new parameter; but since the two parameters M and i constitute a context, we can now say that in CIL expressions are interpreted relative to a context and a world. In fact, where $\mathrm{c} \in \mathrm{Dom}_{\mathcal{M}}(\tau\text{-Mod}) \times \mathrm{Dom}_{\mathcal{M}} i$, we can let $[\![\zeta]\!]_{\mathcal{M},\mathrm{M},\mathrm{i},\mathrm{w}} = [\![\zeta]\!]_{\mathcal{M},\mathrm{c},\mathrm{w}}$.

To put it crudely, the difference between a static and a dynamic model theory is that the latter relationalizes some of the parameters on which interpretation depends in the static case. Here, we are interested in a logic supporting dynamic operators that change contexts to contexts; so the dynamic models should involve relations over contexts. This can be accomplished by relativizing evaluation to a pair of contexts: $[\![\zeta]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}}$ is the value assigned to ζ relative to a world w and a pair of contexts c and c$'$. The semantic definitions of CIL for complex expressions are generalized unchanged to the dynamic case, except for the clause for identity. This is restricted so as to render identities static; that is, identities are not allowed to change the context. (See the discussion of static expressions, below.)

> Where ζ and ξ are expressions of type τ, $[\![\zeta = \xi]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}} = \top$ iff $c = c'$ and $[\![\zeta]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}} = [\![\xi]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}}$.
>
> Where ζ is an expression of type τ, $[\![\lambda x_\sigma(\zeta)]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}} = \mathrm{f}$, where f is the function from $\mathrm{Dom}_{\mathcal{M}}(\sigma)$ to $\mathrm{Dom}_{\mathcal{M}}(\tau)$ such that $\mathrm{f}(\mathrm{d}) = [\![\zeta]\!]_{\mathcal{M}^{\mathrm{d}}/x,\mathrm{c},\mathrm{c}'\mathrm{w}}$.
>
> Where ζ is an expression of type $\langle\sigma,\tau\rangle$ and ξ is an expression of type σ, $[\![\zeta(\xi)]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}} = [\![\zeta]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}}([\![\xi]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}})$.
>
> Where ζ is an expression of type τ, $[\![{}^\wedge\zeta]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}} = \mathrm{f}$, where f is the function from $\mathrm{Dom}_{\mathcal{M}}(w)$ to $\mathrm{Dom}_{\mathcal{M}}(\tau)$ such that $\mathrm{f}(\mathrm{w}') = [\![\zeta]\!]_{M,\mathrm{c},\mathrm{c}',\mathrm{w}'}$.
>
> Where ζ is an expression of type $\langle w,\tau\rangle$, $[\![{}^\vee\zeta]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}} = [\![\zeta]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}}(\mathrm{w})$.
>
> Where ζ is an expression of type τ and $\mathrm{c} = \langle \mathrm{i},\mathrm{M}\rangle$, $[\![{}^\sqcap\zeta]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}} = \mathrm{f}$, where f is the function from $\mathrm{Dom}_{\mathcal{M}}(i)$ to $\mathrm{Dom}_{\mathcal{M}}(\tau)$ such that $\mathrm{f}(\mathrm{i}') = [\![\zeta]\!]_{\mathcal{M},\mathrm{c}'',\mathrm{c}',\mathrm{w}}$ for $\mathrm{c}'' = \langle \mathrm{i}',\mathrm{M}\rangle$.
>
> Where ζ is an expression of type $\langle i,\tau\rangle$ and $\mathrm{c} = \langle \mathrm{i},\mathrm{M}\rangle$, $[\![{}^\sqcup\zeta]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}} = [\![\zeta]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}}(\mathrm{i})$.

To provide a logical environment in which sentences—expressions of type t—can perform changes in the context, we have generalized evaluation to make it depend on a pair of contexts. For uniformity, we have had to treat the interpretation of expressions of arbitrary type in this way, but this generalization doesn't have a natural interpretation for many types; as far as I can see, for instance, although $[\![c_{\langle e,e\rangle}]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}}$ is formally defined, it has no clear intuitive meaning. We need to devise a way of making such cases vacuous. And we must do this in the context of a logic that is not partial, in which $[\![\zeta]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}}$ is always defined.[9]

Consider the type t; here we have some reliable dynamic intuitions. To say that a formula ϕ of type t is dynamically vacuous is to say that ϕ is a test; i.e., that it has no effect on context. Such formulas meet the following condition.

Definition 5.1. *Static type t expressions.*
An expression ϕ of type t is *static in* $\mathcal{M}$ if for all $\mathrm{c}, \mathrm{c}', \mathrm{w}$, if $\mathrm{c} \neq \mathrm{c}'$ then $[\![\phi]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}} = \bot$.

We generalize this definition to arbitrary types by choosing, for each type τ, a designated "null value" $\bot_{\mathcal{M},\tau}$. We make $\bot_{\mathcal{M},t} = \bot$; also, we want to ensure that $\bot_{\mathcal{M},\langle\sigma,\tau\rangle}(\bot_{\mathcal{M},\sigma}) = \bot_{\mathcal{M},\tau}$. We can do this by defining $\bot_{\mathcal{M},\tau}$ by the following induction on types.[10]

Definition 5.2. $\bot_{\mathcal{M},\tau}$.

Basis. $\bot_{\mathcal{M},t} = \bot$; $\bot_{\mathcal{M},e} \in \mathrm{Dom}_{\mathcal{M}}(e)$; $\bot_{\mathcal{M},w} \in \mathrm{Dom}_{\mathcal{M}}(w)$; $\bot_{\mathcal{M},i} \in \mathrm{Dom}_{\mathcal{M}}(i)$.
Induction. $\bot_{\mathcal{M},\langle\sigma,\tau\rangle}$ is the constant function f from $\mathrm{Dom}_{\mathcal{M}}(\sigma)$ to $\mathrm{Dom}_{\mathcal{M}}(\tau)$ such that $\mathrm{f}(\mathrm{d}) = \bot_{\mathcal{M},\tau}$ for all $\mathrm{d} \in \mathrm{Dom}_{\mathcal{M}}(\sigma)$.

An expression of type τ is static in case it receives the type τ null value relative to any pair of different contexts.

Definition 5.3. *Static type* τ *expressions.*
An expression ξ of type τ is *static in* $\mathcal{M}$ if for all $\mathrm{c}, \mathrm{c}', \mathrm{w}$, if $\mathrm{c} \neq \mathrm{c}'$ then $[\![\xi]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}} = \bot_{\tau}$.

A static model is one that treats all constants as static. We can show that in a static model, all expressions are static.

Definition 5.4. *Static model.*
A model $\mathcal{M}$ is *static* if for all constants α, α is static in $\mathcal{M}$.

Theorem 5.1. If $\mathcal{M}$ is static, then all expressions ξ are static in $\mathcal{M}$.

Proof. By induction on the complexity of expressions.

There is a natural correspondence between models of CIL and static models of DCIL. To convert a CIL model $\mathcal{M}$ to an equivalent static DCIL model $\mathcal{M}'$, let $[\![\alpha]\!]_{\mathcal{M}',\mathrm{c},\mathrm{c},\mathrm{w}} = [\![\alpha]\!]_{\mathcal{M},\mathrm{c},\mathrm{w}}$ for all constants α. Use the same identity to convert a static DCIL model to an equivalent CIL model.

We can create special-purpose versions of DCIL with more interesting dynamic properties by adding truthlike constants that exhibit dynamic behavior. I will describe three such examples: ENTER, dynamic conjunction, and a dynamic "else" construction. Also, I will introduce a context-dependent modality [HERE], which selects the necessity operator of the current context.

[HERE] is a constant of type τ-Mod. $[\![\,[\mathrm{HERE}]\,]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}}(\mathrm{p})(\mathrm{w}') = \top$ iff $\mathrm{M}(\mathrm{p})(\mathrm{w}') = \top$, for all $\mathrm{w}, \mathrm{w}', \mathrm{c}, \mathrm{c}'$ and all $\mathrm{p} \in \mathrm{Dom}_{\mathcal{M}}(\langle w, t\rangle)$, where $\mathrm{First}(\mathrm{c}) = \mathrm{M}$.

[9]Providing for partiality in CIL is a very natural step to take, but I believe it is better methodology to take these steps separately and incrementally. I have not yet tried to formulate a partial version of this logic.
[10]Note that the null values of type e, w, and i are chosen arbitrarily. Probably the most natural way to do this would be to create special elements for this purpose.

ENTER is a constant of type $\langle\tau\text{-Mod}, \langle i, t\rangle\rangle$. $[\![\text{ENTER}]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}}(\mathrm{M}, \mathrm{i}) = \top$ iff $c' = \langle M, i\rangle$, for all $\mathrm{c}, \mathrm{c}', \mathrm{M}, \mathrm{i}$.

; is a constant of type $\langle t, \langle t, t\rangle\rangle$. $;(\phi)(\psi)$ is written $\phi\,;\psi$. $[\![\phi\,;\psi]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}} = \top$ iff for some c'', $[\![\phi]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}'',\mathrm{w}} = \top$ and $[\![\psi]\!]_{\mathcal{M},\mathrm{c}'',\mathrm{c}',\mathrm{w}} = \top$.

ELSE is a constant of type $\langle t, \langle t, t\rangle\rangle$. ELSE$(\phi)(\psi)$ is written ϕ ELSE ψ. $[\![\phi\ \text{ELSE}\psi]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}',\mathrm{w}} = \top$ iff either (i) for some c'', $[\![\phi]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}'',\mathrm{w}} = \top$ and $\mathrm{c}' = c''$ or (ii) for all c'' $[\![\phi]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}'',\mathrm{w}} = \bot$ and for some c'', $[\![\psi]\!]_{\mathcal{M},\mathrm{c},\mathrm{c}'',\mathrm{w}} = \top$ and $\mathrm{c}' = \mathrm{c}''$.

These constructs can be used to formalize one of McCarthy's ideas; that the ENTER operator might enable a context to retrieve information from another context by accessing it. The following formalization describes a process that elaborates the database example of Section 4, whereby Ann's database can discover that 'I have a meeting with Ann at 9' holds from the standpoint of Bob's database.

$$\begin{array}{l}\text{ENTER}([\,b\,], b_i)\,;\text{MEET}(I, a, 9)\,;\mathit{ist}([\,b\,], b_i, \text{MEET}(I, a, 9))\\ \quad\text{ELSE}\,\neg\mathit{ist}([\,b\,], b_i, \text{MEET}(I, a, 9); \text{ENTER}([\,a\,], a_i)\end{array}$$

The theory appears to be faithful, on the whole, to McCarthy and Buvač's ideas. However, a dynamic "Exit" operator that returns to the previous context cannot be formalized without making significant changes in the underlying model theory.

I am uncertain myself about the usefulness of operators like ENTER$([\,b\,], b_i)$. I find operators that require explicit names of contexts somewhat unnatural. It may be that, like possible worlds, contexts are best managed by implicit operators. If this is a genuine problem, the entire theory would need to be reworked.

Bear in mind that the material presented in this section is rather tentative. The formal properties of DCIL need to be explored more carefully; and I would very much like to develop sharper intuitions about the interactions between contextual dynamics, higher-order types, and intensionality.

6. Conclusion

There are several dimensions in which the logical framework that I have presented needs to be generalized in order to obtain adequate coverage:

(6.1) The logic needs to be made partial, to account for expressions which simply lack a value in some contexts.

(6.2) To account for default lifting rules, we need a nonmonotonic logic of context.

We have a general sense of what is involved in making a total logic partial, in making a static logic dynamic, and in making a monotonic logic nonmonotonic. For these reasons, I have adopted the strategy of concentrating on how to formulate an appropriate base logic to which these extensions can be made.

There are a number of approaches to the formalization of partial logics; indeed, the main problem with the logic of partiality, it seems to me, is that there are so many alternatives, and it is hard to select between them. Three-valued logic has been used in connection with the logic of context; see [Buvač and Mason, 1993]. However, a four-valued logic is more symmetrical, and plausible arguments, starting with [Belnap, 1977], have been given for its computational usefulness. Most important for the project at hand, [Muskens, 1996] provides an extended study of how to modify IL using this approach to partiality. It is relatively straightforward to adopt Muskens' work to CIL.

There is, however, a much more ambitious application of partiality, according to which indices are regarded not as full, but as partial disambiguations of expressions. This program, which would require a more radical rethinking of the theory, may be needed to deal with applications of context to natural language interpretation, though perhaps it is unnecessary in cases in which indices correspond to carefully constructed knowledge sources. See, for instance, [van Deemter and Peters, 1996] for information on partial disambiguation.

As for nonmonotonicity, although circumscription is usually formulated in second-order extensional logic, it is relatively straightforward to add a theory of circumscription to IL. Similarly, I believe that other standard approaches to nonmonotonic logic could be adapted to a type theoretic framework. But this only provides a bare framework. The problems of developing a nonmonotonic logic capable of formalizing realistic problems in contextual reasoning would remain to be addressed.

None of these logical developments is entirely trivial, and in fact there is material here for many years of work. I hope to report on developments in these directions in the future.

References

[Anderson, 1984] C. Anthony Anderson. General intensional logic. In Dov Gabbay and Franz Guenther, editors, *Handbook of Philosophical Logic, Volume II: Extensions of Classical Logic*, pages 355–385. D. Reidel Publishing Co., Dordrecht, 1984.

[Belnap, 1977] Nuel D. Belnap, Jr. A useful four-valued logic. In J.M. Dunn and G. Epstein, editors, *Modern Uses of Multiple-Valued Logic.* D. Reidel Publishing Co., Dordrecht, 1977.

[Buvač and Mason, 1993] Saša Buvač and Ian Mason. Propositional logic of context. In Richard Fikes and Wendy Lehnert, editors, *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 412–419, Menlo Park, California, 1993. American Association for Artificial Intelligence, AAAI Press.

[Church, 1940] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[Fagin *et al.*, 1995] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge.* The MIT Press, Cambridge, Massachusetts, 1995.

[Fitch, 1952] Frederic B. Fitch. *Symbolic Logic: An Introduction.* The Ronald Press, New York, 1952.

[Gallin, 1975] Daniel Gallin. *Intensional and Higher-Order Logic.* North-Holland Publishing Company, Amsterdam, 1975.

[Giunchiglia *et al.*, 1993] Fausto Giunchiglia, Luciano Serafini, Enrico Giunchiglia, and Marcello Frixione. Non-omniscient belief as context-based reasoning. In Ruzena Bajcsy, editor, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 548–554, San Mateo, California, 1993. Morgan Kaufmann.

[Groenendijk and Stokhof, 1990] Jeroen Groenendijk and Martin Stokhof. Dynamic Montague grammar. Technical Report LP–90–02, Institute for Language, Logic and Information, University of Amsterdam, Faculty of Mathematics and Computer Science, Roeterssraat 15, 1018WB Amsterdam, Holland, 1990.

[Harel, 1984] David Harel. Dynamic logic. In Dov Gabbay and Franz Guenther, editors, *Handbook of Philosophical Logic, Volume II: Extensions of Classical Logic*, volume 2, pages 497–604. D. Reidel Publishing Co., Dordrecht, 1984.

[Kaplan, 1978] David Kaplan. On the logic of demonstratives. *Journal of Philosophical Logic*, 8:81–98, 1978.

[McCarthy and Buvač, 1995] John McCarthy and Saša Buvač. Formalizing context (expanded notes). Available from http://www-formal.stanford.edu/buvac., 1995.

[McCarthy and Buvač, 1998] John McCarthy and Saša Buvač. Formalizing context (expanded notes). In Atocha Aliseda, Rob van Glabbeek, and Dag Westerståhl, editors, *Computing Natural Language*, pages 13–50. CSLI Publications, Stanford, California, 1998.

[McCarthy, 1993] John McCarthy. Notes on formalizing context. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 81–98, Los Altos, California, 1993. Morgan Kaufmann.

[Montague, 1970] Richard Montague. Pragmatics and intensional logic. *Synthèse*, 22:68–94, 1970. Reprinted in *Formal Philosophy*, by Richard Montague, Yale University Press, New Haven, CT, 1974, pp. 119–147.

[Moses, 1988] Yorav Moses. Resource-bounded knowledge. In Moshe Y. Vardi, editor, *Theoretical Aspects of Reasoning About Knowledge: Proceedings of the Second Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 261–276, San Francisco, 1988. Morgan Kaufmann.

[Muskens, 1996] Reinhard Muskens. *Meaning and Partiality.* Cambridge University Press, Cambridge, England, 1996.

[Stone, 1998] Matthew Stone. *Modality in Dialogue: Planning, Pragmatics and Computation.* Ph.d. dissertation, Computer Science Department, University of Pennsylvania, Philadelphia, Pennsylvania, 1998.

[Thomason, 1997] Richmond H. Thomason. Type theoretic foundations for context. In Sasa Buvač and Łucia Iwańska, editors, *Working Papers of the AAAI Fall Symposium on Context in Knowledge Representation and Natural Language*, pages 173–175, Menlo Park, California, 1997. American Association for Artificial Intelligence, American Association for Artificial Intelligence.

[Thomason, 1998] Richmond H. Thomason. Representing and reasoning with context. In Jacques Calmet and Jan Plaza, editors, *Artificial Intelligence and Symbolic Computation; Proceedings of the International Conference AISC'98, Plattsburgh, New York*, pages 29–41, Berlin, 1998. Springer-Verlag.

[Thomason, 1999] Richmond H. Thomason. Type theoretic foundations for context, part 1: Contexts as complex type-theoretic objects. In Paolo Bouquet, Luigi Serafini, Patrick Brézillon, Massimo Benerecetti, and Francesca Castellani, editors, *Modeling and Using Contexts: Proceedings of the Second International and Interdisciplinary Conference, CONTEXT'99*, pages 352–374. Springer-Verlag, Berlin, 1999.

[van Deemter and Peters, 1996] Kees van Deemter and Stanley Peters, editors. *Semantic Ambiguity and Underspecification.* Cambridge University Press, Cambridge, England, 1996.