Epistemic Semantics in Guarded String Models

Eric Campbell
Cornell University
ehc86@cornell.edu

Mats Rooth
Cornell University
mr249@cornell.edu

Abstract

Constructive and computable multi-agent epistemic possible worlds models are defined, where possible worlds models are guarded string models in an epistemic extension of Kleene Algebra with Tests. The account is framed as a formal language Epik (Epistemic KAT) for defining such models. The language is interpreted by translation into the finite state calculus, and alternatively by modeling propositions as lazy lists in Haskell. The syntax-semantics interface for a fragment of English is defined by a categorial grammar.

1 Introduction and Related Work

Linguistic semantics in the Montague tradition proceeds by assigning propositional *semantic values* to disambiguated sentences of a natural language. A proposition is a set or class of *possible worlds*. These are often assumed to be things with the same nature and complexity as the world we occupy (Lewis, 1986). But alternatively, one can work with small idealized models, in order to illustrate and test ideas. To build such models, spaces of worlds and individuals are stipulated as small finite sets, and semantic values of lexical items are constructed as functions or relations from these sets. Such toy or idealized models are useful in research and in teaching, in that it is possible to represent propositions finitely and explicitly, and to calculate with them. The point of this paper is to scale up toy or idealized models to countable sets of worlds, and to constructive and computable modeling of epistemic alternatives for agents. We describe a certain systematic way of defining such models, and illustrate how to apply them in natural language semantics. The focus is on epistemic semantics and clausal embedding. The fundamental move is to identify possible worlds with strings of primitive events, so that propositions are sets of strings. An advantage in this is that it allows for a mathematical description of an algebra of propositions, coupled with a computational representation using either lazy lists of strings, or finite state machines that describe sets of strings.

The approach taken here synthesizes five antecedents in a certain way. John McCarthy's *Situation Calculus* is the source of the idea of constructing possible worlds as event sequences (McCarthy, 1963; Reiter, 2001). The algebraic theory of *Kleene Algebra with Tests* characterizes algebras with elements corresponding to propositions and event types in our application (Kozen, 2001). The models we propose are an epistemic extension of guarded string models for KAT, where a unary operation interpreted as an existential epistemic modality is included for each agent. *Action models* in dynamic epistemic semantics introduce the technique of constructing epistemic models from primitive alternative relations on events, in order to capture the epistemic consequences of perceptual and communicative events (Baltag et al., 1999). This is the basis for our construction of epistemic alternative relations. Literature on *finite state methods in linguistic semantics* has used event strings and sets of event strings to theorize about tense and aspect in natural language semantics (Fernando, 2004, 2007; Carlson, 2009) and to express intensions (Fernando, 2017). Work on *finite state intensiononal semantics* has investigated how to do the semantics of intensional complementation, including indirect questions, in a setting where compositional semantics is expressed in a finite state calculus (Rooth, 2017; Collard, 2018). We adopt this in our syntax-semantics interface for English.

We begin with examples of event-sequence models. *The Elevator.* An elevator moves up and down in a four-story building, with floors numbered in the European fashion as 0,1,2,3. There are primitive events u (the elevator going up one floor), and

d (the elevator going down on floor). In worlds v_1 and v_2, the events shown in (1) transpire. The truth values for English sentences shown in (2) are observed.

(1)	v_1	u	it goes up from 0 to 1
		u	it goes up from 1 to 2
		d	it goes down from 2 to 1
		u	it goes up from 1 to 2
	v_2	u	it goes up from 0 to 1
		u	it goes up from 1 to 2
		u	it goes up from 2 to 3

(2)	v_1	v_2	Sentence
	false	true	It's on floor 3.
	true	true	It has gone up.
	true	false	It has gone down.
	true	false	It could go up.

The Concealed Coin. Amy and Bob are seated at a table. There is a coin on the table under a cup, heads up. The coin could be heads or tails, and neither agent knows which it is. This initial situation is possible world w_1. Two additional worlds w_2 and w_2 are defined by sequencing events after the initial state, with events interpreted as in (3). The truth values for English sentences shown in (5) are observed.

(3)	a_1	Amy peeks at heads, by tipping the cup. Bob sees she's peeking, but not what she sees.
	b_1	Bob peeks at heads.
	a_0	Amy peeks at tails.
	b_0	Bob peeks at tails.

(4) w_1
$w_2 = w_1\, a_1$
$w_2 = w_1\, a_1\, b_1$

(5) w_1	w_2	w_3	Sentence
false	true	true	Amy knows that it's heads.
false	false	true	Bob knows thats it's heads.
false	false	true	Bob knows Amy knows it's heads.
false	true	true	Bob knows Amy knows whether it's heads or tails.

The events in the examples come with pre-conditions. The elevator can not go up if it is already on floor 3, so u has the pre-condition of the elevator being of floor 0, 1, or 2. Similarly d has the precondition that the elevator is on floor 1, 2 or 3. Amy can peek at heads only if the coin is heads up, so a_1 has the precondition of the coin being heads up. Let h be the Boolean proposition that the coin is heads up. In the other example, let q be the proposition that the elevator is on a high floor (2 or 3), and p be the proposition that it is on an odd floor (1 or 3). Then preconditions can be described by Boolean formulas, with h being the precondition of a_1, and $\overline{pq}$ being the precondition of u. Juxtaposition is used for Boolean conjunction, and the overbar for Boolean negation. Events come as well with a relation between prior and following state, for instance with u incrementing the floor. This is expressed using an operator ":" (read "and next") that pairs Boolean formulas. The first line in (6) describes a_1 (Amy looking at heads) as happening only in an h state, and as not changing the state. Symmetrically, a_0 (Amy looking at tails) can happen only in a not-h state, and does not change the state. The third line says that u increments the floor, and can happen only on floors 0, 1, and 2. The fourth line describes d in similar terms. Plus is disjunction.

(6)	a_1	$h : h$
	a_0	$\bar{h}{:}\bar{h}$
	u	$(\bar{q}\bar{p}){:}(\bar{q}p) + (\bar{q}p){:}(q\bar{p}) + (q\bar{p}){:}(qp)$
	d	$(\bar{q}p){:}(\bar{q}\bar{p}) + (q\bar{p}){:}(\bar{q}p) + (qp){:}(q\bar{p})$

2 Epistemic guarded string models

In the discussion at the end of the last section, a sequence such as $\bar{q}p$ can be viewed as a valuation of primitive test propositions, which is used to describe world state. The primitives are listed in fixed order, and left unmarked (indicating true) or marked with the overbar (indicating false). Suppose that in the coin example, we have an additional primitive stative proposition (or atomic test) t, interpreted as tails. Since a coin is heads or tails but not both, we want to allow the valuations $h\bar{t}$ and $\bar{h}t$, and disallow ht and $\bar{h}\bar{t}$. This is enforced by a *state formula*, which is a Boolean formula, in this case the one given on the second line of (7). Where B is a set of atomic tests and ϕ is a state constraint over B, $\mathcal{A}^{\phi}_{\mathsf{B}}$ is the set of valuations of B that make formula ϕ true. Valuations are called atoms, because they correspond to the atoms of a Boolean algebra of tests (Kozen, 2001).

Formulas like the ones in (6) that describe pre- and post-conditions are *effect formulas*. They are state formulas $\qquad (a \in \mathsf{B})$

$$\rho, \sigma, \varphi \quad ::= \quad a \mid 0 \mid 1 \mid \rho + \sigma \mid \rho\sigma \mid \bar{\rho}$$

effect formulas

$$\zeta, \eta \quad ::= \quad \rho : \sigma \mid \zeta + \eta \mid \zeta \,\&\, \eta \mid \bar{\zeta}$$

$$\begin{aligned}
[\![\rho : \sigma]\!]^{\varphi} &= \mathcal{A}_{\mathsf{B}}^{\rho\varphi} \times \mathcal{A}_{\mathsf{B}}^{\sigma\varphi} \\
[\![\zeta + \eta]\!]^{\varphi} &= [\![\zeta]\!]^{\varphi} \cup [\![\eta]\!]^{\varphi} \\
[\![\zeta \,\&\, \eta]\!]^{\varphi} &= [\![\zeta]\!]^{\varphi} \cap [\![\eta]\!]^{\varphi} \\
[\![\bar{\zeta}]\!]^{\varphi} &= \mathcal{A}_{\mathsf{B}}^{\varphi} \times \mathcal{A}_{\mathsf{B}}^{\varphi} \setminus [\![\zeta]\!]^{\varphi}
\end{aligned}$$

Figure 1: Syntax of state formulas and syntax and semantics of effect formulas. Effect formulas denote relations between atoms. In a state formula, juxtaposition $\rho\,\sigma$ is conjunction.

interpreted as defining relations between atoms, as defined in Figure 1. The atoms they relate are constrained by the state formula as well. For the heads-tails example, let the state formula and the effect formula for a_1 (Aly peeking at heads) be as specified in (7). Then $\mathcal{A}_{\mathsf{B}}^{\varphi}$ and the relation on atoms for the event a_1 are as given at the bottom in (7).

(7)	B	$\{h, t\}$
	state formula φ	$h\bar{t} + \bar{h}t$
	effect formula ζ for a_1	$h : h$
	$\mathcal{A}_{\mathsf{B}}^{\varphi}$	$\{h\bar{t}, \bar{h}t\}$
	$[\![\zeta]\!]^{\varphi}$	$\{\langle h\bar{t}, h\bar{t}\rangle\}$

Epik is a specification language for possible worlds models that includes declarations of events and states, state formulas, effect formulas, and additional information. Figure 2 shows an Epik program that describes a possible worlds model for two agents with information about one coin, and events of the agents semi-privately looking at the coin. The line beginning with `state` enumerates B. The line beginning with `constraint` gives the state formula. The lines beginning with `event` declare events and their effect formulas. Finally the lines beginning with `agent` define *event alternative* relations for agents. Each clause with an arrow has a single event symbol on the left, and a disjunction of alternative events on the right of the arrow. The interpretation of Amy's alternatives for b_1 (Bill peeks at heads), is that when b_1 happens, for Amy either b_1 or b_0 (Bill peeks at tails) could be happening.

This paper focuses on defining a concrete possible worlds model from an Epik specification. The

```
state h t
constraint h!t + t!h
event a1 h:h
event a0 t:t
event b1 h:h
event b0 t:t
agent aly
  a1 -> a1
  a0 -> a0
  b1 -> b1 + b0
  b0 -> b1 + b0
agent bob
  b1 -> b1
  b0 -> b0
  a1 -> a1 + a0
  a0 -> a1 + a0
```

Figure 2: Epik program describing a possible-worlds event sequence model for two agents with information about one coin, and events of the agents semi-privately looking at the coin.

models are an extension of guarded-string models for Kleene Algebra with Tests (KAT). This is an algebraic theory that has model classes including guarded string models, relational models, finite models, and matrix models. Our definitions and notation follow (Kozen, 2001). We add syntax and semantics is included to cover multi-agent epistemic semantics.

Guarded strings over a finite alphabet P are like ordinary strings, but with atoms over a set B alternating with the symbols from P. In the algebra described by Figure 2, P is the set of events $\{a_1, a_0, b_1, b_0\}$, and B is $\{h, t\}$. In the elevator example, P is $\{u, d\}$, and B is $\{p, q\}$

Assuming a trivially true state formula ρ, the set of atoms $\mathcal{A}_{\mathsf{B}}^{\rho}$ in the elevator example is $\{\bar{q}\bar{p}$, $\bar{q}p$, $q\bar{p}$, $qp\}$, which we write $\{\hat{0}$, $\hat{1}$, $\hat{2}$, $\hat{3}\}$. In the coin example, as we already saw in (7), $\mathcal{A}_{\mathsf{B}}^{\varphi}$ is $\{h\bar{t}, \bar{h}t\}$, for which we use the shorthand $\{H, T\}$. A guarded string over P and B is a strings of events from P, alternating with atoms over B, and beginning and ending with atoms. (8) gives the encoding as guarded strings of the worlds in (1) and (3). The length of a guarded string p, written $|p|$ is the number of events in p. An atom such as H is a guarded string of length 0.

(8)

World	*Guarded string*	*Length*
v_1	$\hat{0}\,u\,\hat{1}\,u\,\hat{2}\,d\,\hat{1}\,u\,\hat{2}$	4
v_2	$\hat{0}\,u\,\hat{1}\,u\,\hat{2}\,u\,\hat{3}$	3
w_1	H	0
w_2	$H\,a_1\,H$	1
w_3	$H\,a_1\,H\,b_1\,H$	2

The discussion of (4) mentioned building worlds by incrementing worlds with events. This is accomplished in guarded string models with fusion product $\diamond$, a partial operation that combines two guarded strings, subject to the condition that the atom at the end of the the first argument is identical to the atom at the start of the second one. (9) gives some examples.

(9) $\hat{0}\,u\,\hat{1}\,u\,\hat{2}\,d\,\hat{1}\ \diamond\ \hat{1}\,u\,\hat{2} = \hat{0}\,u\,\hat{1}\,u\,\hat{2}\,d\,\hat{1}\,u\,\hat{2}$
$\hat{0}\,u\,\hat{1}\,u\,\hat{2}\,d\,\hat{1}\ \diamond\ \hat{2}\,u\,\hat{3} =$ *undefined*
$H\ \diamond\ H\,a_1\,H = H\,a_0\,H$
$H\ \diamond\ T\,a_1\,T =$ *undefined*

Rather than individual guarded strings, elements of a guarded string model for KAT are sets of guarded strings. In our application, these elements have the interpretation of propositions, which are sets of possible worlds. In a free guarded string model for KAT, any event can be adjacent to any atom in a guarded string that is an element of the underlying set for the algebra. We instead impose the constraints coming from the state and effect formulas. (10) defines the well-formed guarded strings determined by and Epik specification. Condition (i) says that each atom is consistent with the state constraint, and condition (ii) says that each constituent token event $\alpha_i e_i \alpha_{i+1}$ is consistent with the effect constraint on e_i.[1]

(10) Given P, B, a state formula φ, and an effect formula ζ_e for each event e in P, $\alpha_0 e_0 ... e_n \alpha_{n+1}$ is well-formed iff
(i) $\alpha_i \epsilon \mathcal{A}^{\varphi}_{\mathsf{B}}$ $(0 \leq i \leq n)$, and
(ii) $\langle \alpha_i, \alpha_{i+1} \rangle \epsilon [\![\zeta_{ei}]\!]^{\varphi}$, $(0 \leq i \leq n)$.

Well-formed guarded strings have the interpretation of worlds in the application to natural-language semantics. The set of possible worlds in the Kripke frame determined by an Epik specification is the set of well-formed guarded strings. At this point, we could say that any set of worlds is a proposition, so that the set of propositions is the power set of the set of worlds (Montague and Thomason, 1975; Callin, 1975). We will instead define a more restrictive set of propositions corresponding to the regular sets of strings. This is deferred to the next section.

In the natural language application, sets of well-formed guarded strings have the addional interpretation of event types. An event-type is something that can "happen" is different worlds. The event of the elevator going up is modeled not as the bare event symbol u or its unit set $\{u\}$, but as the set of guarded strings $\{\hat{0}\,u\,\hat{1}, \hat{1}\,u\,\hat{2}, \hat{2}\,u\,\hat{3}\}$. The event of the elevator going up in a given world w corresponds to w being incremented to form $w \diamond e$, where e is an element of the event type. (11) gives the event types in the examples.

(11)

	Event type
u	$\{\hat{0}\,u\,\hat{1}, \hat{1}\,u\,\hat{2}, \hat{2}\,u\,\hat{3}\}$
d	$\{\hat{1}\,d\,\hat{0}, \hat{2}\,d\,\hat{1}, \hat{3}\,d\,\hat{2}\}$
a_1	$\{H\,a_1\,H\}$
a_0	$\{T\,a_0\,T\}$
b_1	$\{H\,b_1\,H\}$
b_0	$\{T\,b_0\,T\}$

The construction so far defines a set of worlds from an Epik specification. Normally and in our examples, W is countably infinite, though some choices of effect formulas can result in a finite set of worlds. The next step is to define an alternative relation R_a on worlds for each agent a. This will result in a Kripke frame $\langle W, R_1, ..., R_n \rangle$ consisting of a set of worlds, and a world-alternative relation for each agent (Kripke, 1963). An Epik specification defines an alternative relation on bare events for each agent a, which we notate as R_a. This should be lifted to a relation $\hat{R}_a$ on worlds. The basic idea is that when a world w is incremented with an event e, in the resulting world $w \diamond e$, epistemic alternatives for agent a are of the form $w' \diamond e'$, where w' is an alternative to for a in w, and e' is and event-alternative to e for a.[2] This needs to be im-

[1]An alternative is to define equations such as $\bar{\phi} = 0$ (from the state formula ϕ) and $a_1 = h a_1 h$ (from the effect formula $h : h$ for event a_1), and construct a quotient algebra from the equivalence relation generated by these equations. This results in equating sets of guarded strings in the free algebra that differ by guarded strings that are ill-formed according to the state and effect formulas. In the development in the text, we instead use a set of guarded strings that are well-formed according to the state and effect formulas as the representative of the equivalence class.

[2]In this it is important that the event-alternative relation for an agent is constant across worlds. We anticipate that the definition given here produces results equivalent to what is found in literature on event alternatives in dynamic epistemic semantics, though we have not verified this. That literature primarily focuses on mapping an epistemic model for a single

plemented in a way that takes account of pre- and post-conditions for events. For this, our approach is to refer the definition of well-formed guarded strings. (12) defines a relation on worlds from a relation on bare events.

(12) Let W be a set of guarded strings over events P and primitive tests B, and R be a relation on P. The corresponding relation $\hat{R}$ on W holds between a guarded string

$$\alpha_0 e_0 ... e_n \alpha_{n+1}$$

in W and a guarded string q iff q is an element of W and is of the form

$$\alpha'_0 e'_0 ... e'_n \alpha'_{n+1},$$

where for $0 \leq n$, $\langle e_i, e'_i \rangle \epsilon \hat{R}$.

This requires that in the alternative world q, each constituent event e'_i is an alternative to the event e_i in the base world according to $\hat{R}$. Compatibilities between events in the alternative world are enforced by the requirement that the alternative q is an element of W, so that state and effect formulas are enforced.

Consider a scenario like the one from Figure 1, but with an additional agent Cem. The base world Tb_0Tc_0T is one where the coin is tails, and first Bob looks at tails, and them Cem looks at tails. The first event b_0 has the alternatives b_0 and b_1 for Amy, and the second event c_0 has the alternatives c_0 and c_1 for Amy. This results in four combinations b_0c_0, b_0c_1, b_1c_0, and b_1c_1. But these are filtered by post- and pre-conditions of events in the alternative world, so that the set of alernatives for Amy in Tb_0Tc_0T is $\{Tb_0Tc_0T, Hb_1Hc_1H\}$, with two world-alternatives instead of four.

3 The logical language of Epistemic KAT

In addition to a set of possible worlds and an algebra of propositions, the application in natural language requires a logical language for naming propositions. The standard language for Kleene algebra with tests has the signature $\langle K, +, \cdot, *, ^-, 0, 1\rangle$ (Kozen, 2001). In a guarded string model for KAT,

time and situation to another, and uses general first-order models, rather than guarded string models. See Baltag et al. (1999), Van Ditmarsch et al. (2007), and articles in Van Ditmarsch et al. (2015). We picked up the idea from papers by Moss and his colleagues, together with Moore (1984) and subsequent literature in situation theory. This previous literature is motivated by epistemic logic and AI planning, rather than natural language semantics.

events	$e \in \mathsf{P}$
states	σ as in Figure 1
p, q	$::= e \mid \sigma \mid p+q \mid pq \mid p^* \mid \neg p \mid \Diamond_a p$
$\Box_a p$	$\triangleq \neg\Diamond_a \neg p$
$p \wedge q$	$\triangleq \neg(\neg p + \neg q)$
$\bullet$	$\triangleq \sum_{e \in \mathsf{P}} e$

Figure 3: The language of Epik terms and key derived operators.

K is a set of sets of guarded strings, $+$ is set union, the operation $\cdot$ is fusion product raised to sets, $*$ is Kleene star, the operation $^-$ is complement for tests, 0 the empty set, and 1 is the set of atoms.[3]. To this signature we add a unary modal operation $\Diamond_a$ for each agent, and a unary complement operation $\neg$ on elements of K. The intended interpretation of $\Diamond_a p$ is the set of worlds where proposition p is epistemically possible for agent a. Propositional complement is included because natural languages have sentence negation. In addition, universal box modalities are defined as duals of existential diamond modalities.

With modalities and propositional negation added, the signature of n-agent epistemic KAT is $\langle K, +, \cdot, *, ^-, 0, 1, \neg, \Diamond_1 ... \Diamond_n\rangle$. Figure 3 defines the syntax of the language. Juxtaposition is used in place of the product symbol $\cdot$. Terms in this language are used to represent the proposositional semantic values of English sentences. (13) gives some examples. To explain the first one $\bullet$ as defined in Figure 3 is the disjunction of the primitive events. Since a world is a well-formed sequence of events, $\bullet*$ is the set of worlds. Multiplying by the state symbol h in the term $\bullet * h$ has the effect of conjoining h with the atom at the end of the world. So $\bullet * h$ is the set of worlds where the coin is heads.[4].

(13)

$\bullet * t$	It's tails.
$\bullet * h$	It's heads.
$\Box_a \bullet * h$	Amy knows that it's heads.

$\Box_b(\Box_a \bullet * t + \Box_a \bullet * \neg t)$
Bob knows that Amy knows whether it's tails.

Each term in the language is interpreted as a set

[3]0 has the dual role the identity for $+$ (union), and as False for operations on tests. 1 has the dual role of the identity for product (fusion product raised to sets), and True for tests.

[4]A mapping from English sentences to logical terms in epistemic KAT is presented in Section 6

$$\begin{array}{lcl}
[\![0]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} & \triangleq & \emptyset \\
[\![1]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} & \triangleq & \mathcal{A}^{\varphi}_{\mathsf{B}} \\
[\![b]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} & \triangleq & \mathcal{A}^{b\varphi}_{\mathsf{B}} \\
[\![\bar{\sigma}]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} & \triangleq & \mathcal{A}^{\varphi}_{\mathsf{B}} \setminus [\![\sigma]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} \\
[\![e]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} & \triangleq & \{\alpha e\beta | \alpha\zeta_e\beta\} \\
[\![p+q]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} & \triangleq & [\![p]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} \cup [\![q]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} \\
[\![pq]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} & \triangleq & \left\{ x \diamond y \;\middle|\; \begin{array}{l} x \in [\![p]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} \\ y \in [\![q]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} \\ x \diamond y \text{ is defined} \end{array} \right\} \\
[\![p*]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} & \triangleq & \bigcup_{n\geq 0} [\![p^n]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} \\
[\![\neg p]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} & \triangleq & [\![\bullet *]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} \setminus [\![p]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta} \\
[\![\Diamond_a p]\!] & \triangleq & \{x \mid \exists y. x\hat{R}_a y \wedge y \in [\![p]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta}\}
\end{array}$$

Figure 4: Interpretation of Epik terms as sets of guarded strings (Version M)

of guarded strings. Where p is a term, $[\![p]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta}$ is the interpretation of the term, where superscript captures dependence on an Epik specification. Figure 4 defines the interpretation of terms recursively. The interpretation $[\![1]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta}$ of the multiplicative identity is the set of atoms that satisfy the state constraint φ. Where b is a primitive Boolean, $[\![b]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta}$ is the set of atoms that satisfy the state constraint and where b is true. Where e is a primitive event, $[\![e]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta}$ is the set of guarded strings that have the form of e flanked by compatible atoms, as determined by the event formula ζ_e. The product pq of terms p and q is interpreted with fusion product raised to sets of guarded strings. Kleene star is interpreted as the union of iterated products. p^n is the n-times product of p with itself, with $p^0 = 1$. Propositional complement is complement relative to the set of worlds. The epistemic formula $\Diamond_a p$ is interpreted with Kripke semantics for epistemic modality, as the pre-image of the complement proposition under the relation $\hat{R}_a$. Here $\hat{R}_a$ is the world-alternative for agent a, as defined in (12).

Summing up, given an Epik specification, a state constraint φ, and effect relation ζ_e for each primitive are constructed. Given these, each term p (as defined syntactically in Figure 3) is interpreted as a set of guarded strings $[\![p]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta}$ over the Boolean primitives and event primitives in the specification. Let $K^{\mathsf{B},\mathsf{P},\varphi,\zeta}$ be the sets that are interpretations of terms. Then $\langle K^{\mathsf{B},\mathsf{P},\varphi,\zeta}, +, \cdot, *, \bar{}, 0, 1, \neg, \Diamond_{a_1}, ..., \Diamond_{a_n}\rangle$ is a concrete guarded string interpretation for the signature of epistemic KAT, with operations as in Figure 4

$$\begin{array}{lcl}
[\![0]\!]^{\varphi,\mathcal{E}} & \triangleq & \{\} \\
[\![1]\!]^{\varphi,\mathcal{E}} & \triangleq & \mathcal{A}^{\varphi}_{\mathsf{B}} \\
[\![e]\!]^{\varphi,\mathcal{E}} & \triangleq & \hat{\mathcal{E}}^{\varphi}(e) \\
[\![\psi]\!]^{\varphi,\mathcal{E}} & \triangleq & \mathcal{A}^{\varphi\ \psi} \\
[\![p+q]\!]^{\varphi,\mathcal{E}} & \triangleq & [\![p]\!]^{\varphi,\mathcal{E}} \cup [\![q]\!]^{\varphi,\mathcal{E}} \\
[\![p;q]\!]^{\varphi,\mathcal{E}} & \triangleq & [\![p]\!]^{\varphi,\mathcal{E}} \diamond [\![q]\!]^{\varphi,\mathcal{E}} \\
[\![p^*]\!]^{\varphi,\mathcal{E}} & \triangleq & \bigcup_n \underbrace{[\![p]\!]^{\varphi,\mathcal{E}} \diamond \ldots \diamond [\![p]\!]^{\varphi,\mathcal{E}}}_{n} \\
[\![\neg p]\!]^{\varphi,\mathcal{E}} & \triangleq & [\![\_^*]\!]^{\varphi,\mathcal{E}} \setminus [\![p]\!]^{\varphi,\mathcal{E}} \\
[\![\Diamond_a\, p]\!] & \triangleq & \{x \mid x\, \hat{R}_a\, y, y \in [\![p]\!]^{\varphi,\mathcal{E}}\}
\end{array}$$

Figure 5: Interpretation of Epik terms as sets of guarded strings

(e.g. the binary operation $+$ is union, and the unary operation $\Diamond_a$ is pre-image relative to $\hat{R}_a$). The construction can also be viewed as providing a concrete n-agent Kripke frame $\langle [\![\bullet *]\!]^{\mathsf{B},\mathsf{P},\varphi,\zeta}, \hat{R}_1, ..., \hat{R}_n\rangle$.[5] The frame consists of a set of worlds, and an epistemic-alternative relation for each agent. These models are used as a target for natural-language interpretation in Section 5 and Section 6.

4 Translation into the finite state calculus

The finite state calculus is an algebra of regular sets of strings and regular relations between strings that was designed for use in computationonal phonology and morphographemics (Kaplan and Kay, 1994; Beesley and Karttunen, 2003). Current implementations allow for the definition of functions that have the status of defined operations on regular sets and relations (Hulden, 2009; Karttunen, 2010). Such definitions are used here to construct of a model for epistemic KAT inside the finite state calculus. The space of worlds is a set of ordinary strings. Bit sequences (sequences of 0's and 1's) are used for atoms, and these alternate with event symbols in the encoding of a world. (14) gives the encoding of some of the worlds from the examples. A string is a finite sequence of symbols, and `0`, `1`, `u`, and `d`, are symbols. `a0` and `b0` are multi-character symbols that are found in implementations of the finite state calculus. The multi-character symbol `a0` is an element of the alphabet that has no connection with the element `a`.

(14) Worlds coded as strings

[5]The domain of the Kripke frame differs from the domain of the guarded string model, because the former is the set of worlds, while the latter is the set of propositions.

World	*String*
v_1	`0 0 u 0 1 u 1 0 d 0 1 u 1 0`
v_2	`0 0 u 0 1 u 1 0 u 1 1`
w_1	`1 0`
w_2	`1 0 a1 1 0`
w_3	`1 0 a1 1 0 b1 10`

Terms in the finite state calculus are interpreted as sets of strings, or for relational terms, as relations between strings. Computationally, the sets and relations are represented by finite state acceptors. As used here, a program in the Fst language of the finite state calculus is a straight-line program that defines a sequence of constants naming sets, constants naming relations, and functions mapping one or more regular relations to a regular set or relation. The functions are defined as macros. To illustrate, (15) is part of an Epik program that declares states and a state constraint. This is equivalent to the Fst program (16). The result of the sequence of definitions is that `St` is the set of Boolean vectors or length four that are well-formed according to the state formula; Nst is the complement operation for tests; H1 is the set of well-formed Boolean vectors that have 1 in the first position; and so forth. Translation from an Epik program to an Fst program is accomplished in Haskell.

(15) Part of an Epik program

```
state H1 T1 H2 T2
constraint !(H1 T1) & (H1 + T1)
           !(H2 T2) & (H2 + T2)
```

(16) Corresponding Fst program

```
define Bool ["0"|"1"];
define St0 [Bool Bool Bool Bool];
define H1 [1 Bool Bool Bool];
define T1 [Bool 1 Bool Bool];
define H2 [Bool Bool 1 Bool];
define T2 [Bool Bool Bool 1];
define St St0 & ((((St0 - (H1 & T1))
                & (H1 | T1))
                & (St0 - (H2 & T2)))
                & (H2 | T2));
define Nst(X) [St - X];
define H1 St & H1;
define T1 St & T1;
define H2 St & H2;
define T2 St & T2;
```

Event symbols are interpreted as event types, in the way discussed in connection with 11. For instance, defines an event type `paT2type` of Amy peeking at the second coin in a scenario with two coins, and seeing that it is tails.[6] The first conjunct expresses that this event can only happen if the second coin is tails (`H2`), and that the state of the second coin does not change. The second conjunct expresses that the event does not change the state of the first coin – either it is heads (`H1`) and remains heads, or it is tails (`T1`) and remains tails. Such formulas are mapped from effect formulas in Epik programs. The state formula is imposed as well, since it figures in the definition of state constants such as `H1`.

(17)
```
define paT2type [[[T2 paT2] T2] &
  [[[H1 paT2] H1] | [[T1 paT2] T1]]];
```

Semantically in Fst, `paT2type` is a set of two strings, which can be displayed in an interpreter for the language by reading `paT2type` as a regular expression and printing its extension, see (**??**).

(18)
```
xfst[1]: regex paT2type;
3.0 Kb. 16 states, 16 arcs, 2 paths.
xfst[2]: print words
0 1 0 1 paT2 0 1 0 1
1 0 0 1 paT2 1 0 0 1
```

The finite state calculus has a product operation of string concatenation raised to sets. Concatenation of strings with atoms (Boolean vectors) at both ends has the effect of doubling atoms at the juncture, and does not enforce matching of atoms at the juncture. Therefore KAT product can not be identified with product in the finite state calculus. Instead, KAT product and KAT Kleene star are defined operations, see Figure 6. The binary product operation `Cn` and the unary Kleene star operation `Kst` combine strings in the string algebra, remove strings with non-matching atoms, and then delete the second of two tests to create well-formed guarded strings. Matching of atoms is enforced with the set `Wf0`, which is the set of strings that do not contain non-matching Boolean vectors. The containment operator (expressed by the dollar sign) and complement (expressed by tilde) are operators of the finite state calculus. The set of non-matching sequences of atoms `UnequalStPair` is defined by a finite disjunction. Deletion is accomplished by a re-write rule in the finite state calculus, which is a notation for defining regular relations by contextually constrained substitutions (REFS). In this case, is `Squash` us a regular relation that deletes an atom (a sequence of 0's and 1's of a certain length) when it follows an atom.[7]

[6]After the definition, `paT2type` is a constant symbol denoting a set of two strings. On the right hand side of the definition, `paT2` is the unit set of the symbol `paT2`.

[7]This is a non-equal length regular relation. The finite

`St` Tests such as 0 1 1 0. The length is the number of generators.

`UnequalStPair` Sequence of two unequal tests such as 0 1 1 0 0 1 1 1, differing in one or more positions.

```
define Wf0 ~[$ UnequalStPair];
```

String that doesn't contain a non-matching test pair.

```
define Squash St -> 0 || St _;
```

Rewrite relation deleting the second of two tests.

```
define Cn(X,Y)
  [[[X Y] & Wf0] .o. Squash].l;
```

KAT product.

```
define Kpl(X)
  [[[X+] & Wf0] .o. Squash].l;
define Kst(X) St |  Kpl(X);
```

KAT Kleene plus and Kleene star. The Fst operation `|` is union.

Figure 6: Definition in Fst of KAT product and KAT Kleene star. Where `X` and `Y` are regular sets and `R` and `S` are regular relatiions, `X&Y` is the intersection of `X` and `Y`, `X|Y` is the union of `X` and `Y`, `~X` is the complement of `X`, `R.o.S` is the composition of `R` and `S`, `R.l` is the co-domain of `R`, and `$X` is the set of strings that have a substring in X.

A given bare event such as a_1 (Aly looks at heads) is in the KAT algebra a set of bare events decorated with compatible tests on each side, semantically $\{10a_110\}$ in this case. This is a unit set rather than a guarded string, because elements of the KAT algebra are sets. Worlds in the KAT algebra are defined by sequencing events using *Kst*. The operation enforces compatibility of states, so that $(a_1 + a_0)(b_1 + b_0)$ contains two worlds rather than four. The program in Figure 2 as interpreted in FST defines a countably infinite set of possible worlds by KAT Kleene closure as $\mathit{Kst}(a_1 + a_0 + b_1 + b_0)$, and an algebra of propositions as regular sets of strings drawn from this space of worlds.

It remains to define an epistemic alternative relation on worlds for each agent. The relevant information in Figure 2 is a relation between bare events for each agent. This determines a relation in the guarded string algebra a relation between bare events decorated with compatible tests. For agent

state calculus includes such relations, and they can be used with relation composition and relation domain and co-domain. They are restricted in that the complement and set difference for non-equal length relations is not defined. The epistemic alterative relations that are defined in Figure – are equal-length relations.

$$\texttt{define RelKpl(R)}\ \underbrace{\texttt{Squash.i.o.}}_{c}$$

$$\underbrace{\texttt{Wf0.o.}}_{b}\underbrace{\texttt{[R+]}}_{a}\underbrace{\texttt{.o.Wf0}}_{b}\underbrace{\texttt{.o.Squash}}_{c}$$

a Relational Kleene plus in the string algebra

b Constrain domain and co-domain to contain no unmatched tests.

c Reduce doubled tests to a single test in the domain and co-domain.

```
define Kst(R) [St.x.St]|Kpl(X);
```

The Fst operation `.x.` is Cartesian product. `R.i` is the inverse of relation `R`.

Figure 7: Definition in Fst of the Kleene concatenation closure of a relation between guarded strings.

Aly, this is the relation described in (19) as a set of ordered pairs.

$$(19)\quad \left\{\begin{array}{l} \langle 10a_110, 10a_110\rangle\,, \\ \langle 01a_001, 01a_001\rangle\,, \\ \langle 10b_110, 10b_110\rangle\,, \\ \langle 10b_110, 01b_001\rangle\,, \\ \langle 01b_001, 10b_110\rangle\,, \\ \langle 01b_001, 01b_001\rangle \end{array}\right\}$$

The relation on decorated events needs to be generalized to a relation of worlds. The principle for this is that an epistemic alternative to a world of the form we is a world of the form vd, where v is a world-alternative to w, d is an event-alternative to e, and vd is defined (i.e. the world alternative v satisfies the pre-conditions of the event alternative d). This principle is found in earlier literature (Moore 198x, Baltag, Moss and Solecki 20xx). In the construction in Fst, the definition of world alternatives takes a simple form. Where R_a is the relation on decorated events for agent a, the the corresponding relation on worlds in is the Kleene closure of R_a. Where R and S are relations, the concatenation product of R and S is the set of pairs of the form $\langle x_1x_2, y_1y2\rangle$, where $\langle x_1, y_1\rangle$ is in relation R, and $\langle x_2, y_2\rangle$ is in relation S. The Kleene closure of relation R is $\cup_{n\geq 0}R^n$, where R^n is the n-times concatenation product of R with itself (the 0-times concatenation product is $[\![1]\!]^{\varphi,\mathcal{E}}$). This is an operation in the finite state calculus. Figure **??** defines the corresponding operation in the guarded string algebra. The epistemic alternative relation on worlds for an agent is then defined as the concatenation closure of the event alternative relation for the agent.

Other operations in the guarded string algebra as defined in FST are simpler. Union is union in the string algebra. The complement of a proposition is complement relative to the set of worlds, as defined by set difference in the string algebra.

This scheme provides for an interpretation of the language of Epik terms that is defined in Figure 3. An Epik specifications such as Figure 1 is translated to a straight-line program of the finite state calculus that defines constants and functions, including the ones from Figures 2 and 3. In Xfst or Foma, which are interpreters for the finite state calculus, the program is read, and then propositional terms can be mapped to finite state machines that represent sets of guarded strings, interpreted as propositions.

5 Bounded Lazy Interpretation

We also implement the semantics of Epik terms using lazy lists (**?**) in Haskell, rather than the direct interpretation as sets. Unfortunately, regular expressions, and hence also Epik, can denote infinite sets of strings, for example the term $\_^*$. Normally regular languages are represented using a finite coalgebra (). However the non-distributivity of $\diamond$ accross ; complicates this construction[8]. To sidestep this, we parameterize the interpretation function on a positive integer n and only produce guarded strings of length n or less.

We translate the Epik terms into a Haskell algebraic datatype that represents terms with the same signature as described in Section 2. We then paramaterize the interpretation function, $\mathrm{L}p\mathrm{M}_n^{\varphi,\mathcal{E}}$ on an integer n that describes the maximum length of a string we will produce.

The bounded interpretation into lists of strings is very similar to the unbounded interpretation into sets of strings, except for the bounds checking. The full details are shown in Figure **??**. First note that when $n = 0$, we simply return the empty list, denoted []. Terms of the form 0, 1, e, and ψ have the same denotation as before, translated into a list (for a set X, $\lfloor X \rfloor$ is a list with the same elements as X arbitrarily ordered). The term $p + q$ denotes the list concatenation (written $+\!\!+$) of the denotation of p and of q. The term $p; q$ denotes the fusion product (lifted to lists) of the denotations of p and in q restricted to only those strings shorter than n (for a list l, $l|_n$ filters out elements longer than n). The denotation of p^* uses the fact that p^* and $1 + p; p^*$ are equivalent, and decrements the size threshold on the recursive denotation of p^* by i, where i is the length of the longest (nonzero) string in the denotation of p, making sure to filter out guarded strings that are too long. The denotation of $\neg p$ is the strings that occur in $\_^*$ and not in p (the $\setminus$ operator on lists is analogous to the $\setminus$ operator on sets). The denotation of $\Diamond_a\, p$ is analogous to the set denotation, depicted in Figure **??**-comprehension notation[9].

$$
\begin{array}{rcl}
\mathrm{L}p\mathrm{M}_0^{\varphi,\mathcal{E}} & \triangleq & [] \\
\mathrm{L}0\mathrm{M}_n^{\varphi,\mathcal{E}} & \triangleq & [] \\
\mathrm{L}1\mathrm{M}_n^{\varphi,\mathcal{E}} & \triangleq & \lfloor \mathcal{A}_{\mathsf{B}}^{\varphi} \rfloor \\
\mathrm{L}e\mathrm{M}_n^{\varphi,\mathcal{E}} & \triangleq & \lfloor \hat{\mathcal{E}}^{\varphi}(e) \rfloor \\
\mathrm{L}\psi\mathrm{M}_n^{\varphi,\mathcal{E}} & \triangleq & \lfloor \mathcal{A}_{\mathsf{B}}^{\varphi\ \psi} \rfloor \\
\mathrm{L}p+q\mathrm{M}_n^{\varphi,\mathcal{E}} & \triangleq & \mathrm{L}p\mathrm{M}_n^{\varphi,\mathcal{E}} +\!\!+ \mathrm{L}q\mathrm{M}_n^{\varphi,\mathcal{E}} \\
\mathrm{L}p;q\mathrm{M}_n^{\varphi,\mathcal{E}} & \triangleq & \left(\mathrm{L}p\mathrm{M}_n^{\varphi,\mathcal{E}} \diamond \mathrm{L}q\mathrm{M}_n^{\varphi,\mathcal{E}}\right)|_n \\
\mathrm{L}p^*\mathrm{M}_n^{\varphi,\mathcal{E}} & \triangleq & [] + \left(\mathrm{L}p\mathrm{M}_n^{\varphi,\mathcal{E}} \diamond \mathrm{L}p^*\mathrm{M}_m^{\varphi,\mathcal{E}}\right)|_n \\
\text{where} & & i = \max\{1, \min\{|g| \mid g \in \mathrm{L}p\mathrm{M}_n^{\varphi,\mathcal{E}}\}\} \\
 & & m = \max\{0, n-i\} \\
\mathrm{L}\neg p\mathrm{M}_n^{\varphi,\mathcal{E}} & \triangleq & \mathrm{L}\_^*\mathrm{M}_n^{\varphi,\mathcal{E}} \setminus \mathrm{L}p\mathrm{M}_n^{\varphi,\mathcal{E}} \\
\mathrm{L}\Diamond_a\, p\mathrm{M}_n^{\varphi,\mathcal{E}} & \triangleq & [g' \mid g'\, \hat{R}_a\, g,\ \texttt{for}\ g\ \texttt{in}\ \mathrm{L}p\mathrm{M}_n^{\varphi,\mathcal{E}}]
\end{array}
$$

Figure 8: Bounded interpretation using lazy lists

To convert from a list of guarded strings l to a set of guarded strings, we simply write $\lceil l \rceil$. Note that for any p, φ, $\mathcal{E}$, and n, $\left\lceil \mathrm{L}p\mathrm{M}_n^{\varphi,\mathcal{E}} \right\rceil = \{g \mid g \in [\![p]\!]^{\varphi,\mathcal{E}}, |g| \leq n\}$.

The lists we use here are *lazy* (as opposed to *strict*), which broadly means that computation is delayed until the value is needed. This allows us to avoid computing large, unnecessary iterations. In the elevator example, this means that $\hat{0}; u; u; u^*$ will unroll the u^* 0 times, once, and then discover that it is impossible to unroll it a second time, because $\mathrm{L}\hat{3}; u\mathrm{M}_n^{\varphi,\mathcal{E}} = []$, for any n. So $\mathrm{L}\hat{3}; u\mathrm{M}_n^{\varphi,\mathcal{E}} \diamond \mathrm{L}u^*\mathrm{M}_{n-1}^{\varphi,\mathcal{E}}$ will return [] without having to compute the full fixpoint of $\mathrm{L}u\mathrm{M}^{\varphi,\mathcal{E}}$. Similar behavior occurs once $n = 0$.

Conversely, if we used sets (as in the math) instead of lists we would need to constantly verify the set invariant (that every element is unique) which means processing every element in the set. Verifying that every element in $\mathrm{L}u^*\mathrm{M}^{\varphi,\mathcal{E}}$ is unique would require the full computation of the fixpoint.

[8]Axiomatic and Coalgebraic models for Epik are open questions

[9]List comprehension notation is analogous to set builder notation, except that it is written using square brackets. The order preservation is indicated by the keyword `for`

Basic statives	It's heads. It's tails.
That-complement	Amy knows that it's heads.
Wh-complement	Amy knows whether its heads.
Negation	Bob doesn't know that it isn't heads.
Tensed and base verbal forms	Bob knows that it's heads. Bob doesn't know that it's heads.
Sentence conjunction	It's heads and Bob doesn't know that it's heads
Predicate conjunction	Bob knows that Amy knows whether it's heads and doesn't know that Amy knows that it's heads.

Figure 9: Phenomena covered in the English grammar fragment.

Amy	e	R_a
Bob	e	R_b
it	d	d
heads	$d\backslash_D t$	$\lambda x.0^c{\cdot}h$
tails	$d\backslash_D t$	$\lambda x.0^c{\cdot}!h$
is	$(d\backslash t)/(d\backslash_D t)$	$\lambda P.\lambda x.Px$
knows	$(e\backslash t)/_M t$	$\lambda p.\lambda R.\Diamond Rp$
that	$(((e\backslash t)/_M t)\backslash(e\backslash t))/t$ $\lambda p.\lambda m.\lambda R.{\sim}(m({\sim}p)R)$	
whether	$(((e\backslash t)/_M t)\backslash(e\backslash t))/t$ $\lambda p.\lambda m.\lambda R.{\sim}(m({\sim}p)R)$ $+{\sim}(mpR)$	

Figure 10: Partial categorial grammar lexicon. The first column has a word form. the second column a categorial type, and third column a semantic translation in a logical language that extends the Epik term language with lambda.

6 Syntax-semantics interface

An architecture of interpretation by transtlation is employed, where English sentences are mapped to terms in the logical language via an interpreted grammar, and these terms are in turn interpreted as propositions (sets of possible worlds). For the latter, there are options of translation into the finite state calculus in order to represent propositions as finite state machines (Section 4), and representation in Haskell via lazy lists of guarded strings (Section 5). The grammar is a semantically interpreted multi-modal categorial grammar, consisting of a lexicon of words, their categorial types, and interpretations in a logical lambda language. Figure 9 lists phenomena that are covered. There is recursion through conjuction and verbal complementation, so that the language is infinite. It includes talk of beliefs about beliefs, or in general, talk of arbitrarily iterated belief.

Figure 10 gives illustrative lexical entries.[10] The grammar and semantics are optimized for a simple fragment of English concerned with clausal complementation. The agent names *Amy* and *Bob* contribute the epistemic alternative relations for those agents, rather than individuals. This is possible because the agents are never arguments of extensional predicates, so what matters about the agents is their epistemic alternative relations. The root verb *know* contributes existential modal force. The complementizers *that* and *whether* are the heads of their dominating clauses, and assemble an alternative relation, modal force, and proposition contributed by the complement. These complementizers introduce the dual via two negations, in order to express universal modal force. These moves are offered here as a way of constructing a compact interpreted grammar. They can easily be reformulated in a more comprehensive interpreted grammar of English.

Multimodal categories such as $\backslash_D$ and $\backslash_M$ are used to control the derivation. For instance the category of *heads* is $d\backslash_D t$. The dummy expletive subject *it* has category d. However the phrase *it heads* of category t can not be formed, because $\backslash_D$ is not syntactically active as a function. Instead *it is heads* is formed with a predicator *is* of category $(d\backslash t)/(d\backslash_D t)$. Similarly *knows* has a category with the top-level slash $/_M$, and combines to form a sentence as an argument of *that* or *whether*. These complementizers have a category that looks for the category of *know* on the left, after combining with a complement sentence on the right.

The semantic translations in the third column of Figure 10 use the Epik term language, incremented with lambda. The body of $\lambda x.0^c{\cdot}h$, which is the semantic lexical entry for *heads*, is a term

[10]Category symbols use Lambek/Bar-Hillel notation for slashes, so that $(d\backslash t)/(d\backslash_D t)$ combines with $d\backslash_D t$ on the right to give a value that combines with d on the left to give t.

denoting the set of all worlds where the coin is heads, expressed as the set of all guarded strings that end with a Boolean valuation where the primitive proposition h (it's heads) is true. There is λx at the front because the grammar formalism enforces a strict correspondence between syntactic and semantic types. However, the lambda does not bind anything, because sentences such as *it isn't heads* have an expletive subject. The body $\Diamond Rp$ of $\lambda p.\lambda R.\Diamond Rp$, which is the semantic lexical entry of *knows*, is an Epik term denoting the pre-image of the world-alternative relation contributed by the subject. This is not the right semantics for *Amy knows that it's heads*, because it is an existential modality $\Diamond_R p$, rather than an universal modality $\Box_R p$. This is corrected by the complementizer *that* or *whether*, which introduces the dual.

Sentences are parsed with a chart parser for categorial grammar. The semantics for complex phrases are obtained by syntactic application of semantic translations, accompanied by beta reduction. Semantic terms in the parsing formalism are expressions of untyped lambda calculus. The grammar is set up so that lambda is eliminated by beta reduction in the semantic term corresponding to a sentence. In consequence, the semantic term translating a sentence is a term of the Epik term language. Such a term designates a set of possible words (guarded strings) in the possible worlds model deterimened by an Epik specification. By way of example, (20a) is an English sentence with predicate conjunction and three levels of clausal embedding. Using the grammar and parser, the sentence is mapped to the Epik term (**??**,) which here takes the form of a Scheme s-expression.

Using the result from Section 3, this term can be mapped in an implementation of the finite state calculus to a finite state machine that represents a countably infinite set of possible worlds, represented as guarded strings. Using the result from Section 4, it can be mapped to an infinite lazy list of guarded strings, representing the same set of possible worlds. Either of these is a concrete computational representation of the propositional semantic value ⟦Amy knows that Bob knows that Amy knows whether it is heads and knows that Bob does not know that Amy knows that it is tails⟧^{o}, in the familiar sense of Montague semantics for natural language.

(20) Amy knows that Bob knows that Amy knows whether it is heads and knows that Bob doesnt know that Amy knows that it is tails.

7 Examples and discussion

(21) a. Amy knows that it is tails
b. `(Not (Diamond amy (Not tails)))`

(22) a. Bob doesnt know that Amy knows that it is tails
b. `(Not (Not (Diamond bob (Not (Not (Diamond amy (Not tails)))))))`

(23) a. Bob knows that Amy knows whether it is heads.
b. `(Not (Diamond bob (Not (Or (Not (Diamond amy (Not heads))) (Not (Diamond amy heads))))))`

(24) a. Amy knows that Bob knows that Amy knows whether it is heads and knows that Bob doesnt know that Amy knows that it is tails.
b. `(And (Not (Diamond amy (Not (Not (Diamond bob (Not (Or (Not (Diamond amy (Not heads))) (Not (Diamond amy heads))))))))) (Not (Diamond amy (Not (Not (Not (Diamond bob (Not (Not (Diamond amy (Not tails)))))))))))`

Page breakdown

1,2	3.75	
3	1.25	FST translation
4	1.25	Haskell Epik
5	0.75	English CG fragment
6	1.0	Examples and discussion

This stuff was at the end of Section 2, it became redundant. Also it refers to the interpretation function, which is introduced in Section 3.

(25) $$\hat{R} \triangleq \{\langle x, y\rangle \mid \exists u_1 ... \exists u_n \exists v_1 ... \exists v_n. \\ u_1\ R\ v_1 \wedge ... \wedge u_n\ R\ v_n \wedge \\ x \in [\![u_1; ...; u_n]\!]^{\varphi,\mathcal{E}} \wedge \\ y \in [\![v_1; ...; v_n]\!]^{\varphi,\mathcal{E}}\ \}$$

This defines an epistemic alternative to world x to be a world of the same length, where each component event in the alternative is an event-alternative to the event in corresponding position in the base world. Fusion product enforces preconditions and a correspondence between pre-states and post-states of events on both sides of the epistemic alternative relation. This provides for finitely specifiable construction of epistemic models that reflect intuitions about information exchange and epistemic consequences of perceptual events. See Section 6 for linguistic examples. Since an epistemic alternative has the same lenght as its base world, it follows from the construction that agents know how many events have transpired in their base worlds.

Fortunately, Epik provides a syntax to specify the input-output behavior via the `event` declarations. From these we construct a function $\mathcal{E}$ from agents to relations on atoms that specify the effects each action can have. Notably for every declaration `event` $e\,\zeta$, we have $\mathcal{E}^\varphi(e) = [\![\zeta]\!]^\varphi$. We can lift $\mathcal{E}^\varphi$ to $\hat{\mathcal{E}}$ which decorates the events with their permitted atoms, that is $\hat{\mathcal{E}}^\varphi(e) = \{\alpha\, e\, \beta \mid \alpha, \beta \in \mathcal{E}^\varphi(e)\}$. See the examples in (11).

References

Baltag, A., L. S. Moss, and S. Solecki (1999). The logic of public announcements, common knowledge, and private suspicions.

Beesley, K. R. and L. Karttunen (2003). *Finite State Morphology*. Center for the Study of Language and Inf.

Callin, D. (1975). *Intensional and Higher-order Modal Logic: With Applications to Montague Semantics*. North-Holland Publishing Company.

Carlson, L. (2009). *Tense, Mood, Aspect, Diathesis*. Book ms., University of Helsinki.

Collard, J. (2018). Finite state reasoning for presupposition satisfaction. In *Proceedings of the First International Workshop on Language Cognition and Computational Models*, pp. 53–62.

Fernando, T. (2004). A finite-state approach to events in natural language semantics. *Journal of Logic and Computation 14*(1), 79–92.

Fernando, T. (2007). Observing events and situations in time. *Linguistics and Philosophy 30*(5), 527–550.

Fernando, T. (2017). Intensions, types and finite-state truthmaking. In *Modern Perspectives in Type-Theoretical Semantics*, pp. 223–243. Springer.

Hulden, M. (2009). Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*, pp. 29–32. Association for Computational Linguistics.

Kaplan, R. M. and M. Kay (1994). Regular models of phonological rule systems. *Computational linguistics 20*(3), 331–378.

Karttunen, L. (2010). Update on finite state morphology tools. *Ms., Xerox Palo Alto Research Center*.

Kozen, D. (2001). Automata on guarded strings and applications. Technical report, Cornell University.

Kripke, S. (1963). Semantical considerations on modal logic. *Acta Philosophica Fennica 16*, 83–94.

Lewis, D. (1986). *On the plurality of worlds*, Volume 322. Oxford Blackwell.

McCarthy, J. (1963). Situations, actions, and causal laws. Technical report, Stanford CS.

Montague, R. and R. H. Thomason (1975). Formal philosophy. selected papers of richard montague.

Moore, R. C. (1984). A formal theory of knowledge and action. Technical report, DTIC Document.

Reiter, R. (2001). *Knowledge in Action: Logical foundations for specifying and implementing dynamical systems*. MIT press.

Rooth, M. (2017). Finite state intensional semantics. In *IWCS 2017-12th International Conference on Computational Semantics-Long papers*.

Van Ditmarsch, H., J. Y. Halpern, W. van der Hoek, and B. P. Kooi (2015). *Handbook of Epistemic Logic*. College Publications.

Van Ditmarsch, H., W. van Der Hoek, and B. Kooi (2007). *Dynamic Epistemic Logic*, Volume 337. Springer Science & Business Media.