

Epistemic Semantics in Guarded String Models

Anonymous ACL *Sem submission

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. 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 worlds 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 small 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 on epistemic semantics and clausal embedding. The fundamental move is to identify possible worlds with strings 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 2003). 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 introduced the technique of constructing epistemic models from primitive alternative relations on events, in order to capture the epistemic consequences of perceptual events (Baltag, Moss, and Solecki 1998). 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 2003, Carlson 2009) and to express propositions (Fernando 2017). Literature on finite state intensional semantics has shown how to do the semantics of intensional complementation including indirect questions in a setting where propositions are represented using finite state machines, and 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 primi-

tive events u (the elevator going up one floor), and d (the elevator going down one 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 (H). The coin could be H (heads) or T (tails), and neither agent knows which it is. This initial situation is possible world w_1 . Two additional worlds w_2 and w_3 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 H, by tipping the cup.
Bob sees she's peeking, but not what she sees.
 b_1 Bob peeks at H.

- (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 it's H.
 - false false true Bob knows it's H.
 - false false true Bob knows Amy knows it's H.
 - false true true Bob knows Amy knows whether it's H or T.

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 $!(pq)$ being the precondition of u . Juxtaposition is used for Boolean conjunction, and the exclamation point 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 $(!h):(!h)$
 u $(!qp):(!qp) + (!qp):(qp) + (qp):(qp)$
 d $(qp):(!qp) + (qp):(!qp) + (qp):(qp)$

2 Epistemic guarded string models

Figure 1 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` lists the basic stative propositions. To illustrate syntax, a second proposition t (tails) is included. The line beginning with `constraint` defines compatibilities among the propositions: the coin is heads or tails and not both. The lines beginning with `event` declare events, their preconditions, and their effect on state, following the format in (6). 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.

Kleene Algebra with Tests is an algebraic theory that is defined by equations and inequalities, which has model classes including guarded string models, relational models, finite models, and matrix models. This paper focuses on defining a family of

```

200      state h t
201      constraint h!t + t!h
202      event a1 h:h
203      event a0 t:t
204      event b1 h:h
205      event b0 t:t
206      agent aly
207          a1 -> a1
208          a0 -> a0
209          b1 -> b1 + b0
210          b0 -> b1 + b0
211      agent bob
212          b1 -> b1
213          b0 -> b0
214          a1 -> a1 + a0
215          a0 -> a1 + a0

```

Figure 1: 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.

concrete guarded string algebras, the elements of which are sets of guarded strings. Definitions and notation mostly follow Kozen (2003). Additional syntax and semantics is included to model multi-agent epistemic semantics. Guarded strings over a finite alphabet E are like ordinary strings, but with truth assignments to a set T of primitive propositions (primitive tests) alternating with the symbols from E . In the algebra described by Figure 1, E is the set of events $\{a_1, a_0, b_1, b_0\}$, and in the elevator example, $\{u, d\}$. In the elevator example, T is $\{p, q\}$, and in the coin example it is $\{h, t\}$. We write truth assignments to the primitive propositions that observe the constraint as Boolean vectors. In the coin example, we get the vectors 01, 10 and in the elevator example, the vectors 00, 01, 10, 11. Guarded strings are strings of events, alternating with such vectors, and starting and ending with vectors. (7) gives the encoding as guarded strings of the worlds in (1) and (3).

(7) $v_1 \quad 00u01u10d01u10$
 $v_2 \quad 00u01u10u11$
 $w_1 \quad 01$
 $w_2 \quad 01a_001$
 $w_3 \quad 01a_001b_001$

The discussion of (4) mentioned building worlds by incrementing worlds with events. This is accomplished in guarded string models with fusion prod-

uct, a partial operation that combines two guarded strings, subject to the condition that the truth assignment at the end of the the first argument is identical to the truth assigment at the start of the second one. (8) gives some examples.

(8) $00u01u10d01 \cdot 01u10 = 00u01u10d01u10$
 $00u01u10d01 \cdot 10u11 = \text{undefined}$
 $01 \cdot 01a_001 = 01a_001$
 $01 \cdot 10a_110 = \text{undefined}$

Rather than guarded strings, elements of a guarded string model for KAT are sets of guarded strings. In the application, these elements have the interpretation of propositions (sets of possible worlds) and/or event types. An event such as u in the guarded string model corresponds to the set of guarded strings where the bare event is flanked by compatible truth assigments, as defined in an Epik program by the event declaration and the constraint declaration. These have the status of event types, in that they can “happen” in different possible worlds. Here happening corresponds to incrementing the world with some compatible element of the event type. Where e is a bare event symbol, \hat{e} is the corresponding set of guarded strings, consisting of the bare event decorated with compatible truth assignments. See the examples in (9).

(9)	e	\hat{e}
u	$\{00u01, 01u10, 10u11\}$	278
d	$\{01d00, 10d01, 11d10\}$	279
a_1	$\{10a_110\}$	280
a_0	$\{01a_001\}$	281
b_1	$\{10b_110\}$	282
b_0	$\{01b_001\}$	283

Kleene Algebra with Tests has the algebraic signature $\langle K, +, \cdot, *, \bar{\cdot}, 0, 1 \rangle$. To this we add an unary operation \diamond_a for each agent, and a unary complemt operation \cdot^c on propositions. This results in the signature $\langle K, +, \cdot, *, \bar{\cdot}, 0, 1, \cdot^c, \diamond_a, \diamond_b \rangle$ for epistemic KAT with two agents. In a guarded string model in our construction, the elements of K are sets of guarded strings. The operation $+$ is set union. The constant 0 is the empty set. The constant 1 is the set of all truth assignments obeying the constraint on truth assignments, i.e. $\{10, 10\}$ in the coin example. The operation \cdot is fusion product raised to sets: $x \cdot y$ is the set of all defined fusion products of an element of x with and element of y . The operation $*$ is Kleene star, with $x^* = \cup_{i \geq 0} x^i$, where x^i is the i -times product of x with itself and $x^0 = 1$.

Subsets of 1 are also elements of K , and these form the Boolean algebra of tests as a set Boolean algebra. The overbar operation is complement in this Boolean algebra. The complement operation c is complement at the level of sets of guarded strings, with $x^c = 0^c - x$, where the operation on the right is set difference. The epistemic modality \diamond_a is interpreted using Kripke semantics, as pre-image relative to a fixed relation R_a between guarded strings, $\diamond_a x = \{u | \exists v. v \in x \wedge u R_a v\}$. Here u and v are guarded strings, while x is an element of K .

It remains to define the Kripke relation on guarded strings from an agent specification as in Figure 1. An agent specification pairs each bare event with a set of bare events, and so determines a relation between bare events, call it relation R_a for an agent e . This determines a relation \hat{R}_a between decorated events, see (10). It is generalized to a relation \hat{R} on arbitrary guarded strings by relational Kleene star, as defined in (11). The operation \cdot in the definition is KAT product, which enforces matching of tests. The number of terms n is construed as satisfying $n \geq 1$. *What is desired for $n = 0$?*

$$(10) \quad \hat{R} =_{\text{def.}} \{\langle u, v \rangle | \exists c \exists d. c R d \wedge u \in \hat{c} \wedge v \in \hat{d}\}$$

$$(11) \quad \hat{R} =_{\text{def.}} \{\langle x, y \rangle | \exists u_1 \dots \exists u_n \exists v_1 \dots \exists v_n. \\ u_1 \hat{R} v_1 \wedge \dots \wedge u_n \hat{R} v_n \wedge \\ x = u_1 \cdot \dots \cdot u_n \wedge \\ y = v_1 \cdot \dots \cdot v_n\}$$

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 length as its base world, it follows from the construction that agents know how many events have transpired in their base worlds.

Universal box modalities are defined as duals, $\square_a x = (\diamond_a x^c)^c$. For instance, Aly is certain that the coin is heads if and only if she does not consider it possible that it is not heads.

Summing up, given an Epik program with n

agents, we construct a concrete guarded string model $\langle K, +, \cdot, *, \bar{\cdot}, 0, 1, \diamond_i, \hat{R}_i, \dots, \hat{R}_n \rangle$. The elements of K are interpreted as propositions. 0^c is the set of worlds, and it may be countably infinite. \diamond_i is an epistemic modality for the i th agent. Or referring to the Kripke relations \hat{R}_i , the construction defines a multi-agent Kripke frame $\langle 0^c, \hat{R}_1, \dots, \hat{R}_n \rangle$ (usually a countable one) from an Epik specification. 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, where we obtain semantic values such as $\llbracket \text{Amy knows that it's heads and Bob knows that Amy knows whether it is heads or tails, and does not know that it's heads} \rrbracket$ as elements of K . Concretely the propositions are sets of guarded strings (usually countable ones), construed as sets of worlds as they figure in possible worlds semantics for natural language.

3 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 computational phonology and morphographemics (cite Karttunen etc.) Current implementations allow for the definition of functions with the status of defined operators on regular sets and relations. Such definitions are used to define an embedding of epistemic KAT in a string algebra. The methodology follows Section 2 closely. Let \mathcal{K} be an epistemic algebra as described in Section 2. A given element of \mathcal{K} is represented in the string algebra by the very same set of strings, i.e. by a set of strings that have the form of a sequence of bare event symbols, with interleaved Boolean vectors. Product in the KAT can not be modeled as concatenation in the string algebra, because this would not enforce identity of states, and would result in lengthening Boolean vectors at the concatenation point. Instead, KAT product and KAT Kleene star are defined operations in the string algebra, see Figure 2. The operations concatenate in the string algebra, delete strings with non-matching tests, and then delete the second of two tests create a well-formed guarded string.

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

```

400 St Tests such as 0 1 1 0. The length is the number
401 of generators.
402 UnequalStPair Sequence of two unequal
403 tests such as 0 1 1 0 0 1 1 1, differing in one
404 or more positions.
405 define Wf0 ~[$ UnequalStPair];
406 String that doesn't contain a non-matching test
407 pair.
408 define Squash St -> 0 || St _;
409 Rewrite relation deleting the second of two tests.
410 define Cn(X, Y)
411   [[X Y] & Wf0] .o. Squash].l;
412 KAT product in Fst, where & is intersection, .o.
413 is relation composition, and .l is relation image.
414 define Kpl(X)
415   [[[X+] & Wf0] .o. Squash].l;
416 define Kst(X) St | Kpl(X);
417 KAT Kleene plus and Kleene star in Fst. The Fst
418 operation | is union.

```

Figure 2: Translation into Fst of KAT product and KAT Kleene star.

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 1 as interpreted in FST defines a countably infinite set of possible worlds by KAT Kleene closure as $Kst(a_1 + a_0 + b_1 + b_0)$, 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 1 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 Aly, this is the relation described in (12) as a set of ordered pairs.

$$(12) \quad \left\{ \begin{array}{l} \langle 10a_110, 10a_110 \rangle, \\ \langle 01a_001, 01a_001 \rangle, \\ \langle 10b_110, 10b_110 \rangle, \\ \langle 10b_110, 01b_001 \rangle, \\ \langle 01b_001, 10b_110 \rangle, \\ \langle 01b_001, 01b_001 \rangle \end{array} \right\}$$

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

```

define RelKpl(R) Squash.i.o.
c
Wf0.o. [R+] .o. Wf0.o. Squash
b a b 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.
Squash.i is the inverse
of Squash.

```

```

define Kst(R) [St.x.St] | Kpl(X);
The Fst operation .x. is Cartesian product.

```

Figure 3: Definition in Fst of the Kleene concatenation closure of a relation between guarded strings.

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 $\langle x_1x_2, y_1y_2 \rangle$, 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. 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 4. 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

```

500 agents    $a, b \dots$     $a ::= \langle \text{atomic agents} \rangle$ 
501 tests     $T, S \dots$     $T ::= \langle \text{atomic tests} \rangle \mid 0 \mid 1 \mid$ 
502            $T + S \mid T ; S \mid !T$ 
503 propositions    $p, q \dots$     $p ::= \langle \text{atomic events} \rangle \mid$ 
504            $T \mid p+q \mid p\&q \mid$ 
505            $p;q \mid p^* \mid \sim p \mid$ 
506            $\diamond ap$ 

```

Figure 4: The language of Epik terms. Propositional terms are interpreted as sets of guarded strings.

that represent sets of guarded strings, interpreted as propositions.

4 Interpretation using lazy lists of strings

5 Syntax-semantics interface

An architecture of interpretation by translation 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 3), and representation in Haskell via lazy lists of guarded strings (Section 4). The grammar is a semantically interpreted multimodal categorial grammar, consisting of a lexicon of words, their categorial types, and interpretations in a logical lambda language. (??) lists that are covered.

As illustrated towards the end, there is recursion through conjunction and verbal complementation, so that the language is infinite, and includes talk of beliefs about beliefs, or in general, talk of arbitrarily iterated belief.

(13) gives illustrative lexical entries. The grammar and semantics are in certain way 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 arrive universal modal force. These

Basic statives	It's heads. It's tails.	550 551
That-complement	Amy knows that it's heads.	552 553
Wh-complement	Amy knows whether its heads.	554 555
Negation	Bob doesn't know that it isn't heads.	556 557
Tensed and base verbal forms	Bob knows that it's heads. Bob doesn't know that it's heads.	558 559 560 561
Sentence conjunction	It's heads and Bob doesn't know that it's heads	562 563
Predicate conjunction	Bob knows that Amy knows whether it's heads and doesn't know that Amy knows that it's heads.	564 565 566 567 568 569

Figure 5: Phenomena covered in the English grammar fragment.

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.

(13) Lexicon (partial)

Amy	e	R_a	579 580
Bob	e	R_b	581
it	d	d	582
heads	$d \setminus_D t$	$\lambda x.0^c.h$	583
tails	$d \setminus_D t$	$\lambda x.0^c.!h$	584
is	$(d \setminus t) / (d \setminus_D t)$	$\lambda P. \lambda x. Px$	585
knows	$(e \setminus t) / M t$	$\lambda p. \lambda R. \diamond R p$	586
that	$((e \setminus t) / M t) \setminus (e \setminus t) / t$	$\lambda p. \lambda m. \lambda R. \sim (m(\sim p) R)$	587
whether	$((e \setminus t) / M t) \setminus (e \setminus t) / t$	$\lambda p. \lambda m. \lambda R. \sim (m(\sim p) R)$	588
		$+ \sim (mpR)$	589

Multimodal categories such as \setminus_D and \setminus_M are used to control the derivation. For instance the category of *heads* $d \setminus_D t$. The dummy expletive subject *it* has category d , but the phrase *it heads* of category t can not be formed, because \setminus_D is not syntactically active as a function. Instead *it is heads* can be formed with a predicator *is* of category $(d \setminus t) / (d \setminus_D t)$. (This uses Lambek/Bar-Hillel notation for

slashes, so that $(d \setminus t) / (d \setminus_D t)$ combines with $d \setminus_D t$ on the right to give a value that combines with d on the left to give 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*, which has 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 (13) use the Epik term language, incremented with lambda. The body of $\lambda x.0^c.h$, which is the semantic lexical entry for *heads*, is a term denoting the set of all worlds where the coin is heads, represented 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 of a correspondence the grammar formalism uses a correspondence between syntactic and semantic types. However, it does not bind anything, because sentences such as *it isn't heads* have an expletive subject. The body $\Diamond R p$ of $\lambda p. \lambda R. \Diamond R p$, which is the semantic lexical entry of *knows*, is an Epik term denoting the pre-image of the set of worlds p according to the relation \hat{R} between guarded strings that is determined by the event-level relation R . This is not the right semantics for *Amy knows that it's heads*, because it is an existential modality \Diamond_{Rp} , rather than an universal modality \Box_{Rp} . 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 language ($-$). Such a term designates a set of possible words (guarded strings) in the possible worlds model determined by an Epik specification such as the one in Figure 1. (14a) 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 (??). 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 $\llbracket \text{Amy knows that Bob knows that Amy knows whether it is heads and knows that Bob does not know that Amy knows that it is tails} \rrbracket^o$, in the familiar sense of Montague semantics for natural language.

nite 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 $\llbracket \text{Amy knows that Bob knows that Amy knows whether it is heads and knows that Bob does not know that Amy knows that it is tails} \rrbracket^o$, in the familiar sense of Montague semantics for natural language.

- (14) a. 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.

b.

6 Examples and discussion

Page breakdown

1,2	3.75		668
3	1.25	FST translation	669
4	1.25	Haskell Epik	670
5	0.75	English CG fragment	671
6	1.0	Examples and discussion	672