3 Finite state automatons in information states*

Emil Gergely Dyekiss
Eötvös Loránd University

The aim of this paper is to introduce a certain kind of information state representation in a dynamic system of propositional logic, using finite state automatons and highlight its advantages, including relation to inquisitive semantics and belief revision.

1 Preface

Dynamic semantics offers a very straightforward and plausible theory for modelling dialogues. In dynamic semantics the meaning of a sentence is a function which assigns information states to information states. Sentences change the information in the mind of the hearer. (Veltman 1996: 1)

The aim of modelling a dialogue is at least twofold. One is better understanding the process of interactions in dialogues, the other is simulation (or computational implementation) of a dialogue system. In the latter case it is very important what kind of tools and structures the model relies on. In a theory some structures are plausible and easy to use, but they are unusable in an implemented system. Such structures are the infinite sets. It is better to get rid of them already in the theory.

The simplest dynamic semantic theories are modelling information states as model sets. See for instance (Kálmán and Rádai 2001: 88). An information state consists of models assumed possible by the hearer. Depending on the theory, these sets can be infinite. Of course, we cannot handle them in an implemented dialogue system. We cannot store them, just perhaps give a method to enumerate them. It is unlikely that people have such structures in their mind.

If we switch from model sets to structured information states, because of the poor abilities of the former, the latter can be based upon (parts of) formulae to be able to handle revision. But if the dialogue changes without growing information, the information states (based on formulae) change, what contradicts our intuition that without new information, no information state change should be performed. If

*I would like to thank the valuable help of László Kálmán, Márton Makrai and everyone who made comments on my thoughts or asked me about details and finally driving me on a path to arrive to this article — including Márta Maleczki, Márton Muntág, Péter Rebrus, Anna Szabolcsi, Dániel Vásárhelyi. I hope that I did not forget to mention anybody and I admit that all mistakes in this article are my own.

we model information states by automatons then we can perhaps define information states such a way that old information is accepted by the automaton and causes no change in the information state. So perhaps automatons offer a better approach.

In this article I suggest modelling information states by tables and automatons instead of model sets and I investigate the advantages of this turn. I try to apply as simple automatons as possible: finite state automatons, covering as wide variety of phenomena as possible.

2 The language we are modelling

The definitions in this section are based on (Kálmán and Rádai 2001: 207–209, 88-89), but slightly changed.

2.1 Syntax

Definition 1 *The Language of Propositional Logic*

$$L_0 =_{def} \langle LC, NLC, F \rangle$$

Members: Logical and non-logical constants, formulae respectively.

Definition 2 *Logical Constants of* L_0

$$LC =_{def} \{(,), \neg, \wedge, \vee\}$$

Members: Symbols of opening and closing parentheses, negation, conjunction, disjunction, respectively.

Definition 3 *Non-logical constants of* L_0

$$NLC =_{def} \{p, q, ...\}$$

Members: Symbols of atomic propositions.

Definition 4 *Formulae of* L_0

F is the smallest set satisfying the following conditions:

1. *If* $p \in NLC$ *then* $p \in F$. *A propositional constant is a formula.*
2. *If* $A \in F$ *then* $\neg A \in F$. *A negated formula is a formula.*
3. *If* $A, B \in F$ *then* $(A \wedge B) \in F$. *Conjunctive formulae.*
4. *If* $A, B \in F$ *then* $(A \vee B) \in F$. *Disjunctive formulae.*

2.2 Classical Semantics

For later reference, I define the classical semantics for this language.

Definition 5 *Discourse Universe*
$U =_{def} \{T \cup F\}$ *where* T *is the set of true,* F *is the set of false statements and they satisfy that* $T \cap F = \emptyset$ *and* $T \cup F \neq \emptyset$.

Definition 6 *Interpretation Function*
For an interpretation function ρ *it is true that* $\rho(p) \in U$ *in case of* $p \in NLC$.

Definition 7 *Models of Propositional Logic*
$\mathbf{M} =_{def} \langle T, F, \rho \rangle$ *where* T *is the set of true,* F *is the set of false statements and* ρ *is an interpretation function.*

Definition 8 *Classical Semantic Values for Propositional Logic*

1. $[p]^M =_{def} 1$, *if* $\rho(p) \in T$, *0 otherwise.*
2. $[\neg p]^M =_{def} 1$, *if* $\rho(p) \in F$, *0 otherwise.*
3. $[(A \wedge B)]^M =_{def} 1$, *if* $[A] = 1$ *and* $[B] = 1$, *0 otherwise.*
4. $[(A \vee B)]^M =_{def} 1$, *if* $[A] = 1$ *or* $[B] = 1$, *0 otherwise.*

assuming that the semantic value of the formula A in the model M (of language L_0*) is noted by* $[A]^M$ *and* $p \in NLC$; $A, B \in Form$, *furthermore 1 denotes the true, and 0 the false truth value.*

2.3 Simple Update Semantics

Let's take a look at a simple update semantics for this language.

Definition 9 *Information States in Simple Update Semantics*
The set of information states is Σ_0 *and if* $\mathbf{M}$ *is the class of models of language* L_0, *then* $\Sigma_0 =_{def} \mathscr{P}(\mathbf{M})$, *the powerset of* $\mathbf{M}$.

An information state contains the models assumed possible according to the hearer's knowledge.

Definition 10 *Semantic Values in Simple Update Semantics*
If A is a formula, then its semantic value is $[\![A]\!] : \Sigma_0 \to \Sigma_0$

1. $[\![p]\!](\sigma) =_{def} \{M \in \sigma : [p]^M = 1\}$
2. $[\![\neg A]\!](\sigma) =_{def} \sigma \setminus [\![A]\!](\sigma)$
3. $[\![(A \wedge B)]\!](\sigma) =_{def} [\![B]\!]([\![A]\!](\sigma)) = [\![A]\!] \circ [\![B]\!](\sigma)$
4. $[\![(A \vee B)]\!](\sigma) =_{def} [\![A]\!](\sigma) \cup [\![B]\!](\sigma)$

Assuming that $p \in NLC$; $A, B \in Form$ *and* $[A]$ *is the classical semantic value of A.*

Reverting from Contradictory States is Impossible

Note that according to this simple semantics, all formulae eliminate models from the former information state, keeping no models in the information state after a contradiction. Since there is no method to add models to an information state by any formulae, it is not possible to revert from a contradictory information state.[1] Even if there would be a formula with the ability of adding models to an information state, in the case of such a simple structure as a model set, we have no information about which models to add after revision. So we could not get an adequate result after revision.

3 Automatons In Information States

Let's create automatons accepting model sets! For this purpose we have to slightly change the definition of L_0. The only necessary change affects the set of non-logical constants. Non-logical constants should be ordered. I will use a notation of a letter and a number instead of different letters.

Definition 11 *Nonlogical Constants Revisited*
$NLC =_{def} \{p_i : i \in \mathbb{N}$, $p_1 \in NLC$, *and there is no such* $1 < j \in \mathbb{N}$, *that if* $p_j \in NLC$, *then* $p_{j-1} \notin NLC$, *that is: the elements of NLC are such* p_i*-s, that i starts from* 1 *and continuously increases by 1 (maybe until infinity).* $\mathbb{N}$ *denotes the set of natural numbers.*

Now we can create a code for each model, and use this code as input for the automatons. We use the classical semantics of the models.

[1] In the system proposed in my paper we can have several contradictory states. In the original update semantics in (Veltman 1996: 8) there was only one, and it was called the 'absurd state'

Definition 12 *Coding the Models*
The code of a model $M \in \mathbf{M}$ *is a string on the alphabet* $\Sigma = \{0,1\}$. *The* n^{th} *letter in the code is the value of* $[p_n]^M$.

Automatons will read the codes of the models and accept or reject them as necessary. The intention is that such an automaton should accept the models assumed possible for the hearer in the current information state.

The ordering of atomic propositions seems implausible if we think of human behaviour. People have access to propositions so fast, that a random access seems more probable than a sequential one. This is a shadow on my proposal, but helps defining a simple but powerful representation.

About The Finiteness of the Codes

Depending on the size of *NLC*, codes of the models can be of even infinite length. This will not cause problems in practice, because we will construct automatons such a way, that if the automaton enters an accepting (terminal) state, reading the 'rest' of the code will not be necessary, it will be accepted anyway. This terminal state will be reached after finite steps and no edge will lead to other state from it, it will be a 'tale' of the automaton with loopback edges of all letters of the alphabet.

3.1 Examples

Before the exact definition of building automatons for the information states, I will give examples to show how they work.

The automatons on figure 1 contain exactly one initial state (marked by a short arrow pointing towards them[2]) and one terminal state (marked by a short arrow pointing away from them) in the mentioned 'tale' style. The automaton on the left accepts codes starting with '1' what stands for evaluating p_1 to true. No path to the terminal state for codes starting with '0'. The one on the right will reject codes containing '1' as the second letter, i.e. the codes of the models evaluating p_2 to true.

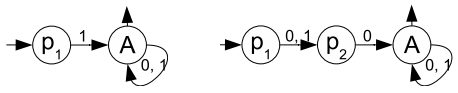

Figure 1: *Two simple automatons: one for* p_1 *and another for* $\neg p_2$

[2]Using the notation of (Eilenberg 1974: 13)

The automaton on figure 2 demonstrates conjunction. There is no edge from the state labelled p_2 to the state labelled p_3 with letter '1', similarly, no edge from p_5 with letter '0'. So this automaton accepts only models evaluating p_2 to false *and* p_5 to true.

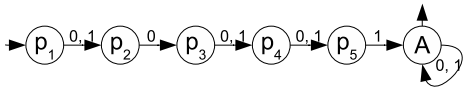

Figure 2: *Automaton for* $\neg p_2 \wedge p_5$

The single(!) automaton on figure 3 has two initial states, and two terminal states. It can be split to two separate branches — each containing one initial and one terminal state. The branches can be treated as separate automatons accepting formulae standing on the sides of the disjunction. First branch accepts models evaluating $(\neg p_2 \wedge p_5)$ to true, the other accepts models evaluating $(p_1 \wedge p_3)$ to true. Finally, the automaton with the two branches accepts models acceptable by the first *or* the second branch.

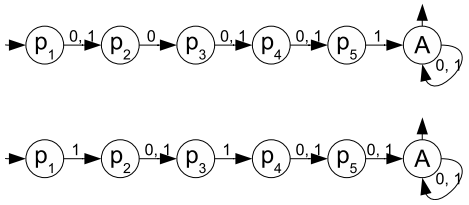

Figure 3: *Automaton including disjunction* $(\neg p_2 \wedge p_5) \vee (p_1 \wedge p_3)$

3.2 Towards Technical Specification

Now it is time to define how we can build these automatons. We have to give exact specification of adding states, edges and how to label them. The last example showed that labelling the states is not so easy. The automaton contained states with similar labels. If we assume several branches, we can get unreadable or confusing, perhaps ambiguous labels. Somehow we have to specify which state we are talking about. This can be achieved by labelling or by some other technique. I choose the latter and introduce a rich description of automatons by tables or spreadsheets. This is defined in the next chapter. I will offer a way for deriving the automatons from these tables.

4 Spreadsheet Semantics for Dialogues

The tables used for representing dialogues are designed for containing a bit more data than necessary for the construction of automatons. They store the whole history of the semantics of the dialogue.

I assume that the table has a header which is not treated as a data row. A table will have some 'administrative' columns additional to the ones containing data for edges. Because of the small alphabet used, tables will be completely different from the usual transition matrix representation. See (Eilenberg 1974: 14). The administrative columns are the following:

Definition 13 *Administrative Columns of the Tables*

Number*: A sequence number for identifying the rows. Topmost row has the number 1 and each row has the number we get by adding one to the number of the row right above it.*

Parent*: The number of the row which was the immediate ancestor of the row.*

Alive*: A value indicating if this row is 'alive' or 'dead' (is participating in the creation of the automaton or not). This value can be 1 (true: alive) or 0 (false).*

The rest of the columns will belong to constants of *NLC*. They will contain nothing or '0' or '1' or both, depending on the edges starting from the state. See definition later.

Now I define the empty table which corresponds to the original, ignorant information state of the hearer.[3]

Definition 14 *Table for the Empty Information State*

It contains only the three administrative column headers, no data rows, no other columns.

Definition 15 *The Effect of an Atomic Formula p_i on the Table*

1. *Changing the Columns*

   We have to achieve to have data columns with header containing p_1 ... p_i from left to right. If there is no column with header p_i, we have to add columns(s) and insert p_n in the header of the header of the n^{th} data column.

2. *Changing the Rows*

   If the table contains only the header row, then we have to add a new row under the header. Its number will be 1, parent 0, alive: 1. In all non-administrative

[3]This state is called 'minimal' in (Veltman 1996: 2)

columns we should write '0, 1'. Finally we have to remove '0' from the column of p_i, keeping only the '1' there.

If the table contained data rows, then we have to copy all of them which are alive, and paste under the last row of the table. We have to change the number of the newly inserted rows to the number which is the result of adding 1 to the number of the row one above the new row. Their parents will be the rows which were copied, and the parent rows should be marked as not alive. Now we have to modify the living rows of the table. If they contain no data in the column of p_i, then we have to fill all the new empty cells of the row by '0, 1'. Finally, we remove '0' from the column of p_i (if it contains '0' at all).

Note that it might happen that a cell in a living row of the table is empty. This means that the branch of the automaton created by using this row is cut into two disconnected parts. The rightmost part will be unreachable, and left hand part will not lead to a terminal state. This case shows that the branch is contradictory.

Definition 16 *The Effect of a Negated Atomic Formula $\neg p_i$ on the Table*
We do the same as in case of an atomic formula, but we remove'1' instead of '0' where appropriate.

Definition 17 *The Effect of a Conjunctive Formula $(A \wedge B)$ on the Table*
First we apply A, then B on the result of the previous operation.

Definition 18 *The Effect of a Disjunctive Formula $(A \vee B)$ on the Table*
If the table does not contain data rows, then we add rows by processing A, then we fill another empty table by processing B. If they contain different number of columns, we add columns (and fill their headers) to the table which contains fewer columns, to get tables with the same number of columns. During this addition we fill the new cells in the living rows of the table by '0, 1'. Finally we merge the two tables by inserting the rows of the table of B under the rows of the table of A. We also have to increase the numbers in the inserted rows by the count of the original rows of the table of A in the columns 'Number' and 'Parent' (except if column 'Parent' contains 0).

If the table already contained data, we have to do the same, but we have to keep the currently living original rows of the table as rows not alive, above the rows generated by A, which will be also above the rows generated by B.

Definition 19 *The Effect of a Negated Conjunctive Formula $\neg(A \wedge B)$ on the Table*
We use De Morgan's law and proceed with processing $(\neg A \vee \neg B)$.

Definition 20 *The Effect of a Negated Disjunctive Formula $\neg(A \vee B)$ on the Table*
We use another law of De Morgan and proceed with processing $(\neg A \wedge \neg B)$.

Definition 21 *The Effect of a Double Negated Formula $\neg\neg A$ on the Table*
We drop the double negation and process A.

#	Parent	Alive	p_1	p_2	p_3	p_4	p_5	*Comment: Header*
1	0	0	0, 1	0				$\neg p_2$
2	1	1	0, 1	0	0, 1	0, 1	1	$(\neg p_2 \wedge p_5)$
3 (*1*)	0	0	1					p_1
4 (*2*)	3 (*1*)	1	1	0, 1	1	*0, 1*	*0, 1*	$(p_1 \wedge p_3)$ + *2 cells!*

Table 1: *The table representation of $(\neg p_2 \wedge p_5) \vee (p_1 \wedge p_3)$ with additional comments*

We defined the effect of all kind of formulae on the tables — but our goal is to have automatons. Now I define how to generate automatons from the tables.

Definition 22 *Generating Automatons Based on Tables*

1. *We care only for the 'living' rows.*
2. *Let's add an automaton state for all cells of the table in the non-administrative columns of data rows.*
3. *Let's mark the states belonging to the first data column as initial states.*
4. *Let's add one more state on the right of the state belonging to the last data column of each data row, marked as terminal state.*
5. *Let's add edges to the automaton. The numbers '0' and '1' in the cells are to be labels for the edges between the cell's automaton state and its right neighbour. If the cell contains both of them, two edges are necessary, but if the cell is empty, we do not have any edges between the two states.*
6. *Let's add edges beginning at the terminal states of the automaton. We need two for each. One with the label '0', and one with '1', both ending also at this state.*

To see the relation between the table representation and the automaton generated from it, compare figure 3 and the table above.

Notes

Note that we use finite state automatons.

If the automaton has only one living row, then the automaton is deterministic, but if it has more than one living rows, then it is non-deterministic. Non-determinism is due to disjunction.

The difference between the table representation and the automatons is the lack of dialogue history in the latter.

The algorithm of automaton generation from the tables is simple. Almost reversible — except the data in the administrative columns and in not living rows — so we cannot reconstruct the history of the dialogue based on the automaton of the information state.

5 Central Semantic Concepts

We have to define a few semantic concepts to be able to say something about the logical properties of the system. We need at least the following:

Definition 23 *Compatibility*

An information state σ is compatible with the formula A if by applying the formula to σ we get an information state with an automaton in which there is at least one path from at least one of the initial states to at least one of the terminal states.

Definition 24 *Incompatibility*

An information state σ is incompatible with a formula A if by applying the formula to the information state we get an information state with an automaton in which there is no path from any of the initial states to any of the terminal states.

Definition 25 *Support*

An information state σ supports a formula A if in the table representation of the information state we get by applying A to σ, every living row of the original table will be a parent of at least one of the new rows of the new table.

Definition 26 *Consequence*

A formula B is consequence of the formulae $A_1, A_2, \ldots, A_n$ iff for every information state σ it is true that by applying A_1 to σ, and then A_2 on its result etc. until A_n, the resulting information state supports B.

6 Outlook

6.1 Inquisitive Semantics

Inquisitive semantics (Groenendijk and Roelofsen 2009) makes difference between the informative and the inquisitive content of sentences. Informative content can eliminate some of the still possible models, while inquisitive content creates (perhaps overlapping) groups of them, called alternatives. The traditional semantics of questions (Groenendijk and Stokhof 1997) uses disjunct partitions of the models — overlapping of the alternatives is the real new light in inquisitive semantics.

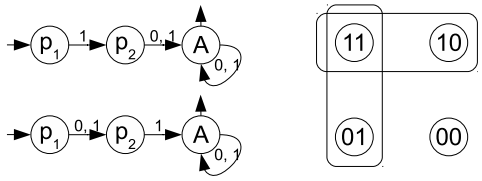

Figure 4: $(\neg p_1 \vee p_2)$ *by an automaton and in inquisitive semantics*

Besides questions, disjunction also bears inquisitive content. In a simple $p_1 \vee p_2$ case inquisitive semantics assumes two alternatives: one in which p_1 is true and another, where p_2 is true. They overlap: p_1 and p_2 can be true at the same time. We can see in figure 4, that the automaton has two branches — they are logically equivalent to the alternatives of inquisitive semantics.

This example is very simple. If we consider more complex alternatives by more complex formulae, then we will have more branches then alternatives. But we can make groups of the branches. Parents of the branches (more precisely: the rows of the table from which we derived the branch) can create the real alternatives from branches. If all the branches have the same parent, a branch alone represents an alternative. If there are more then one branches, and they have (at least some) different parents, then the parents determine the alternatives.

6.2 Belief Revision

The simple update semantics for propositional logic, which represents information states as model sets, is not able to get out of a contradictory information state, because it is represented by an empty set and we do not know which models to add, moreover we do not have operators which can add models to the set (just eliminating rules).

We can define contraction and revision by the table and the automaton representation of information states. The most important application of revision is perhaps to get out of a contradictory information state. A contradictory information state is represented by an automaton in which there is no path from any initial state to any terminal state — or represented by a table which has at least one empty cell in each of its living rows.

In my approach, revision (Alchourrón, Gärdenfors, and Makinson 1985) of an atomic formula or its negation is quite straightforward, because we have to add a '0' or a '1' into each cell in the column which has this atomic formula in its header. (Or in a more complicated way: adding new rows, and making changes there.) This approach enables to get out of the contradictory state by a sequence of some (can be more than one) steps.

Contraction or revision of compound formulae is a bit more difficult. If we want to enable contraction of only those formulae which were part of the dialogue, we need the table representation (opposed to the automaton representation), because it preserves the whole history of the dialogue.

I cannot define the exact steps of contraction here, but it is necessary to note, that the structured information state described in this article make it possible to define contraction at all, which is important.

I also note that questions can make it possible to determine which propositions to contract, and the information state representations described here make it possible to handle questions and answers in general (similarly to disjunction, using the concept of alternatives of inquisitive semantics) and to propose appropriate question for contraction.

7 Summary

In this article I suggested tables for the representation of information states in a dynamic propositional logic system. These tables can store historical data about the dialogue and serve as base for the derivation of finite state automatons which can accept model sets by reading their codes.

I also stated relations to other theories (inquisitive semantics and belief revision) emphasizing that this kind of representation is powerful enough for defining the semantics of questions in line with inquisitive semantics, and for defining contraction and revision, two important operations of belief revision.

Future stories

The system I suggested was sketched briefly and could be defined in a more exact formal way, including the semantics of questions and answers, handling contradiction, contraction and revision. The most useful improvement of this approach would be to extend the theory to predicate logic in which dynamic semantics brings its very natural and impressive power of handling existential quantification and binding of discourse referents.

Bibliography

Alchourrón, C. E., P. Gärdenfors, and D. Makinson (1985). On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic 50*, 510–530.

Eilenberg, S. (1974). *Automata, Languages, and Machines*. New York and London: Academic Press.

Groenendijk, J. and F. Roelofsen (2009, May 6-8). Inquisitive semantics and pragmatics. In *Proceedings of the International Workshop on Semantics, Pragmatics and Rhetorics*, Donostia, Spain.

Groenendijk, J. and M. Stokhof (1997). Questions. *Handbook of Logic and Language 25*, 1055–1124.

Kálmán, L. and G. Rádai (2001). *Dinamikus szemantika*. Budapest: Osiris Kiadó.

Veltman, F. (1996). Defaults in update semantics. *Journal of Philosophical Logic 25*, 221–261.