Undecidability of Boolean bunched logic

James Brotherston

Programming Principles, Logic and Verification Group
Dept. of Computer Science
University College London, UK
J.Brotherston@ucl.ac.uk

Logic Summer School, ANU, 10 December 2015
Previously:

- we showed that the Hilbert system for BBI is sound and complete w.r.t. validity in an associated class of Kripke models;
Previously:

- we showed that the Hilbert system for BBI is sound and complete w.r.t. validity in an associated class of Kripke models;

- we reformulated the Hilbert system as an analytic, cut-eliminating display calculus.
Previously:

- we showed that the Hilbert system for BBI is sound and complete w.r.t. validity in an associated class of Kripke models;

- we reformulated the Hilbert system as an analytic, cut-eliminating display calculus.

You might think that BBI is therefore decidable: given a formula $A$, just conduct an exhaustive search for $\vdash A$ in the display calculus.
Previously:

- we showed that the Hilbert system for **BBI** is sound and complete w.r.t. validity in an associated class of Kripke models;

- we reformulated the Hilbert system as an analytic, cut-eliminating display calculus.

You might think that **BBI** is therefore **decidable**: given a formula $A$, just conduct an exhaustive search for $\vdash A$ in the display calculus.

But, actually, it isn’t. That’s today’s subject.
Recall:

**Provability in BBI** is given by extending a Hilbert system for propositional classical logic by

\[
\begin{align*}
A \ast B &\vdash B \ast A \\
A \ast (B \ast C) &\vdash (A \ast B) \ast C \\
A \vdash A \ast I \\
A \ast I &\vdash A
\end{align*}
\]

\[
\begin{align*}
A_1 \vdash B_1 &\quad A_2 \vdash B_2 \\
A_1 \ast A_2 &\vdash B_1 \ast B_2 \\
A \ast B &\vdash C \\
A &\vdash B \ast C
\end{align*}
\]

\[
\begin{align*}
A &\vdash B \ast C \\
A \ast B &\vdash C
\end{align*}
\]
Recall:

A BBI-model is given by \( \langle W, \circ, E \rangle \), where

- \( W \) is a set (of “worlds”),
- \( \circ \) is a binary function \( W \times W \rightarrow \mathcal{P}(W) \); we extend \( \circ \) to \( \mathcal{P}(W) \times \mathcal{P}(W) \rightarrow \mathcal{P}(W) \) by
  \[
  W_1 \circ W_2 = \text{def } \bigcup_{w_1 \in W_1, w_2 \in W_2} w_1 \circ w_2
  \]
- \( \circ \) is commutative and associative;
- the set of units \( E \subseteq W \) satisfies \( w \circ E = \{w\} \) for all \( w \in W \).

A valuation for BBI-model \( M = \langle W, \circ, E \rangle \) is a function \( \rho \) from propositional variables to \( \mathcal{P}(W) \).
Given $M$, $\rho$, and $w \in W$, we define the forcing relation $w \models_\rho A$ by induction on formula $A$:

\[
\begin{align*}
w \models_\rho P & \iff w \in \rho(P) \\
w \models_\rho A \rightarrow B & \iff w \models_\rho A \text{ implies } w \models_\rho B \\
\vdots \\
w \models_\rho I & \iff w \in E \\
w \models_\rho A \ast B & \iff w \in w_1 \circ w_2 \text{ and } w_1 \models_\rho A \text{ and } w_2 \models_\rho B \\
w \models_\rho A \otimes B & \iff \forall w', w'' \in W. \text{ if } w'' \in w \circ w' \text{ and } w' \models_\rho A \\
& \text{ then } w'' \models_\rho B
\end{align*}
\]

A is valid in $M$ iff $w \models_\rho A$ for all $\rho$ and $w \in W$.  

5/ 21
There is basically one universal strategy for showing things undecidable: by \textit{reduction} from some problem \textit{already} known undecidable!
Undecidability strategy

- There is basically one universal strategy for showing things undecidable: by reduction from some problem already known undecidable!

- That is, we show that if we could decide validity of BBI-formulas, then we could decide some other undecidable problem.
Undecidability strategy

• There is basically one universal strategy for showing things undecidable: by reduction from some problem already known undecidable!

• That is, we show that if we could decide validity of BBI-formulas, then we could decide some other undecidable problem.

• Classic undecidable problem: the halting problem, as famously considered by Turing.
Undecidability strategy

- There is basically one universal strategy for showing things undecidable: by reduction from some problem already known undecidable!

- That is, we show that if we could decide validity of BBI-formulas, then we could decide some other undecidable problem.

- Classic undecidable problem: the halting problem, as famously considered by Turing.

- Turing machines are not very convenient for our purposes (why not?), so we shall instead consider the halting problem for two counter Minsky machines.
Minsky machines

A Minsky machine $M$ with counters $c_1, c_2$ is given by a finite set of labelled instructions of the following types, where $k \in \{1, 2\}$:
Minsky machines

A Minsky machine $M$ with counters $c_1$, $c_2$ is given by a finite set of labelled instructions of the following types, where $k \in \{1, 2\}$:

$L_i: c_k++; \text{goto } L_j; \quad \text{“increment } c_k \text{ (and jump)”}$

$L_i: c_k--; \text{goto } L_j; \quad \text{“decrement } c_k \text{ (and jump)”}$

$L_i: \text{if } c_k = 0 \text{ goto } L_j; \quad \text{“zero-test } c_k \text{ (and jump)”}$

$L_i: \text{goto } L_j; \quad \text{“jump”}$
Minsky machines

A Minsky machine $M$ with counters $c_1$, $c_2$ is given by a finite set of labelled instructions of the following types, where $k \in \{1, 2\}$:

- $L_i$: $c_k ++$; \textbf{goto} $L_j$; “increment $c_k$ (and jump)”
- $L_i$: $c_k --$; \textbf{goto} $L_j$; “decrement $c_k$ (and jump)”
- $L_i$: \textbf{if} $c_k = 0$ \textbf{goto} $L_j$; “zero-test $c_k$ (and jump)”
- $L_i$: \textbf{goto} $L_j$; “jump”

Configurations of $M$ have the form $\langle L_i, n_1, n_2 \rangle$. We write $\langle L_i, n_1, n_2 \rangle \downarrow_M$ if $\langle L_i, n_1, n_2 \rangle \rightsquigarrow^*_M \langle L_0, 0, 0 \rangle$. 
Minsky machines

A Minsky machine $M$ with counters $c_1$, $c_2$ is given by a finite set of labelled instructions of the following types, where $k \in \{1, 2\}$:

- $L_i: c_k++; \text{ goto } L_j; \quad \text{“increment } c_k \text{ (and jump)”}$
- $L_i: c_k--; \text{ goto } L_j; \quad \text{“decrement } c_k \text{ (and jump)”}$
- $L_i: \text{if } c_k = 0 \text{ goto } L_j; \quad \text{“zero-test } c_k \text{ (and jump)”}$
- $L_i: \text{goto } L_j; \quad \text{“jump”}$

Configurations of $M$ have the form $\langle L_i, n_1, n_2 \rangle$. We write $\langle L_i, n_1, n_2 \rangle \downarrow_M$ if $\langle L_i, n_1, n_2 \rangle \leadsto^*_M \langle L_0, 0, 0 \rangle$.

We introduce special labels $L_{-1}, L_{-2}$ with instructions:

- $L_{-1}: c_2--; \text{ goto } L_{-1}; \quad L_{-1}: \text{goto } L_0;$
- $L_{-2}: c_1--; \text{ goto } L_{-2}; \quad L_{-2}: \text{goto } L_0;$

whence $\langle L_{-k}, n_1, n_2 \rangle \downarrow_M$ iff $n_k = 0$. 


Outline proof of undecidability

Theorem
It is undecidable whether a given Minsky machine terminates from a given configuration.
Outline proof of undecidability

**Theorem**

*It is undecidable whether a given Minsky machine terminates from a given configuration.*

**Idea:** given a machine $M$ and configuration $C$, we encode $M, C$ as a formula $\mathcal{F}_{M,C}$ of BBI such that

$$M \text{ terminates from } C \Leftrightarrow \mathcal{F}_{M,C} \text{ is valid}.$$
Outline proof of undecidability

Theorem
It is undecidable whether a given Minsky machine terminates from a given configuration.

Idea: given a machine $M$ and configuration $C$, we encode $M, C$ as a formula $F_{M,C}$ of BBI such that

$$M \text{ terminates from } C \iff F_{M,C} \text{ is valid}.$$ 

Then, if we could decide validity of formulas in BBI, we could decide the halting problem for Minsky machines, contradiction!
Encoding configurations (1)

First, for each label $L_i$ we have a propositional variable $l_i$. 
First, for each label $L_i$ we have a propositional variable $l_i$.

We also pick two propositional variables $p_1$, $p_2$ to represent the counters $c_1$, $c_2$. 
Encoding configurations (1)

First, for each label $L_i$ we have a propositional variable $l_i$.

We also pick two propositional variables $p_1, p_2$ to represent the counters $c_1, c_2$.

Then, a configuration $\langle L_i, n_1, n_2 \rangle$ will be represented as:

$$l_i \ast p_1^{n_1} \ast p_2^{n_2}$$

where $p_k^n$ denotes the formula $\underbrace{p_k \ast p_k \ast \cdots \ast p_k}_{n \text{ times}}$, with $p_k^0 = I$. 
Now we pick a new propositional variable \( b \) and write

\[
- A \overset{\text{def}}{=} A \rightarrow \neg b
\]
Now we pick a new propositional variable $b$ and write

$$-A \overset{\text{def}}{=} A \rightarrow^* b$$

$b$ will be interpreted as “all terminating configurations of the machine”.


Now we pick a new propositional variable $b$ and write

$$-A =_{\text{def}} A \rightarrow \top b$$

$b$ will be interpreted as “all terminating configurations of the machine”.

So $-A$ should be read as “whenever I add $A$ to my current state, I get a terminating configuration”.
Restricted $\ast$-contraction

Contraction does not hold for $\ast$:

$$A \not\vdash A \ast A$$
**Restricted \(*\)-contraction**

**Contraction** does not hold for \(*\):

\[
A \nvdash A \ast A
\]

However, a **restricted** form of contraction does hold:

\[
I \land A \vdash (I \land A) \ast (I \land A)
\]

Easy to see semantically, but quite hard to derive!
Encoding machines in BBI

We code each instruction $\gamma$ of a machine $M$ as a formula $\kappa(\gamma)$ of BBI:

We code a whole machine $M = \{\gamma_1, \ldots, \gamma_t\}$ as:

Finally, we code termination from $\langle L_0, 0, 0 \rangle$ as $(I \land l_0)$.
Encoding machines in BBI

We code each instruction $\gamma$ of a machine $M$ as a formula $\kappa(\gamma)$ of BBI:

$L_i: c_k++; \text{goto } L_j; \implies (- (l_j * p_k) -* -l_i)$

We code a whole machine $M = \{\gamma_1, \ldots, \gamma_t\}$ as:

$\kappa(M) = I \land \bigwedge_{i=1}^{t} \kappa(\gamma_i)$

Finally, we code termination from $\langle L_0, 0, 0 \rangle$ as $(I \land l_0)$. 
Encoding machines in BBI

We code each instruction \( \gamma \) of a machine \( M \) as a formula \( \kappa(\gamma) \) of BBI:

\[
\begin{align*}
L_i: c_k & \quad ; \quad \text{goto } L_j; \quad \Rightarrow \quad \left( -(l_j \ast p_k) \ast -l_i \right) \\
L_i: c_k & \quad ; \quad \text{goto } L_j; \quad \Rightarrow \quad \left( -l_j \ast -(l_i \ast p_k) \right)
\end{align*}
\]
Encoding machines in BBI

We code each instruction \( \gamma \) of a machine \( M \) as a formula \( \kappa(\gamma) \) of BBI:

\[
L_i: c_k++; \text{ goto } L_j; \quad \Rightarrow \quad (- (l_j * p_k) \rightarrow -l_i)
\]
\[
L_i: c_k--; \text{ goto } L_j; \quad \Rightarrow \quad (- l_j \rightarrow (l_i * p_k))
\]
\[
L_i: \text{if } c_k = 0 \text{ goto } L_j; \quad \Rightarrow \quad (-(l_j \lor l_{-k}) \rightarrow -l_i)
\]
We code each instruction $\gamma$ of a machine $M$ as a formula $\kappa(\gamma)$ of BBI:

$L_i: c_k++; \text{ goto } L_j; \Rightarrow (-(l_j \ast p_k) \ast -l_i)$

$L_i: c_k--; \text{ goto } L_j; \Rightarrow (-l_j \ast -(l_i \ast p_k))$

$L_i: \text{ if } c_k = 0 \text{ goto } L_j; \Rightarrow (- (l_j \lor l_{-k}) \ast -l_i)$

$L_i: \text{ goto } L_j; \Rightarrow (-l_j \ast -l_i)$
Encoding machines in BBI

We code each instruction $\gamma$ of a machine $M$ as a formula $\kappa(\gamma)$ of BBI:

- $L_i: c_k++; \ \text{goto} \ L_j; \ \Rightarrow \ (-(l_j \ast p_k) \ast -l_i)$
- $L_i: c_k--; \ \text{goto} \ L_j; \ \Rightarrow \ (-l_j \ast -(l_i \ast p_k))$
- $L_i: \text{if } c_k = 0 \ \text{goto} \ L_j; \ \Rightarrow \ (-(l_j \lor l_{-k}) \ast -l_i)$
- $L_i: \text{goto} \ L_j; \ \Rightarrow \ (-l_j \ast -l_i)$

We code a whole machine $M = \{\gamma_1, \ldots, \gamma_t\}$ as:

$$\kappa(M) = I \land \bigwedge_{i=1}^{t} \kappa(\gamma_i)$$
Encoding machines in BBI

We code each instruction $\gamma$ of a machine $M$ as a formula $\kappa(\gamma)$ of BBI:

\[
\begin{align*}
    L_i: & \ c_k++; \ \textbf{goto} \ L_j; \quad \Rightarrow \quad -(l_j \ast p_k) \ast -l_i \\
    L_i: & \ c_k--; \ \textbf{goto} \ L_j; \quad \Rightarrow \quad -(l_j \ast -(l_i \ast p_k)) \\
    L_i: & \ \textbf{if} \ c_k=0 \ \textbf{goto} \ L_j; \quad \Rightarrow \quad -(l_j \lor l_{-k}) \ast -l_i \\
    L_i: & \ \textbf{goto} \ L_j; \quad \Rightarrow \quad -(l_j \ast -l_i)
\end{align*}
\]

We code a whole machine $M = \{\gamma_1, \ldots, \gamma_t\}$ as:

\[
\kappa(M) = I \land \bigwedge_{i=1}^{t} \kappa(\gamma_i)
\]

Finally, we code termination from $\langle L_0, 0, 0 \rangle$ as $(I \land -l_0)$. 

Putting everything together, the formula $F_{M,C}$ encoding termination of $M$ from $C$ will be

$$\kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land -l_0) \vdash b$$
**Master encoding**

Putting everything together, the formula $F_{M,C}$ encoding termination of $M$ from $C$ will be

$$\kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land -l_0) \vdash b$$

Plan of proof:

$M$ terminates from $C$
Master encoding

Putting everything together, the formula $F_{M,C}$ encoding termination of $M$ from $C$ will be

$$\kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land \neg l_0) \vdash b$$

Plan of proof:

$M$ terminates from $C$

$\Rightarrow F_{M,C}$ provable \hspace{1cm} (Theorem 1)
Master encoding

Putting everything together, the formula $\mathcal{F}_{M,C}$ encoding termination of $M$ from $C$ will be

$$\kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land \neg l_0) \vdash b$$

Plan of proof:

$M$ terminates from $C$

$\Rightarrow \mathcal{F}_{M,C}$ provable \hspace{1cm} (Theorem 1)

$\Rightarrow \mathcal{F}_{M,C}$ valid in all models \hspace{1cm} (soundness)
Putting everything together, the formula $F_{M,C}$ encoding termination of $M$ from $C$ will be

$$\kappa(M) \times l_i \times p_1^{n_1} \times p_2^{n_2} \times (I \land -l_0) \vdash b$$

Plan of proof:

$M$ terminates from $C$

$\Rightarrow F_{M,C}$ provable \hspace{1cm} (Theorem 1)

$\Rightarrow F_{M,C}$ valid in all models \hspace{1cm} (soundness)

$\Rightarrow F_{M,C}$ valid in a specially chosen model and valuation
Putting everything together, the formula $F_{M,C}$ encoding termination of $M$ from $C$ will be

$$
\kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land \neg l_0) \vdash b
$$

Plan of proof:

$M$ terminates from $C$

$\Rightarrow$ $F_{M,C}$ provable \hspace{1em} (Theorem 1)

$\Rightarrow$ $F_{M,C}$ valid in all models \hspace{1em} (soundness)

$\Rightarrow$ $F_{M,C}$ valid in a specially chosen model and valuation

$\Rightarrow$ $M$ terminates from $C$ \hspace{1em} (Theorem 2)
Theorem
Suppose $\langle L_i, n_1, n_2 \rangle \downarrow_M$. Then the following is derivable in BBI:

$$\kappa(M) * l_i * p_1^{n_1} * p_2^{n_2} * (I \land -l_0) \vdash b$$
First theorem

Theorem
Suppose $\langle L_i, n_1, n_2 \rangle \downarrow_M$. Then the following is derivable in BBI:

$$\kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land l_0) \vdash b$$

We actually derive the stronger

$$\kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \vdash \neg \neg l_0$$
First theorem

Theorem
Suppose $\langle L_i, n_1, n_2 \rangle \Downarrow_M$. Then the following is derivable in $\text{BBI}$:

$$\kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land -l_0) \vdash b$$

We actually derive the stronger

$$\kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \vdash \neg\neg l_0$$

Proof is by induction on the length of the computation $\langle L_i, n_1, n_2 \rangle \Downarrow_M$. Restricted $\ast$-contraction is used to duplicate instructions from $\kappa(M)$ as needed.
Choosing a model

Given that $F_{M,C}$ is provable, it is valid by soundness.
Choosing a model

Given that $\mathcal{F}_{M,C}$ is provable, it is valid by soundness.

It’s enough to show that $M$ terminates from $C$ given only that $\mathcal{F}_{M,C}$ is valid in some model of our choice, under some valuation of our choice.
Choosing a model

Given that $\mathcal{F}_{M,C}$ is provable, it is valid by soundness.

It’s enough to show that $M$ terminates from $C$ given only that $\mathcal{F}_{M,C}$ is valid in some model of our choice, under some valuation of our choice.

We use the **RAM-domain model** $\langle D, o, \{e_0\} \rangle$, where:

- $D$ is the set of all finite subsets of $\mathbb{N}$;
Choosing a model

Given that $\mathcal{F}_{M,C}$ is provable, it is valid by soundness.

It’s enough to show that $M$ terminates from $C$ given only that $\mathcal{F}_{M,C}$ is valid in some model of our choice, under some valuation of our choice.

We use the **RAM-domain** model $\langle \mathcal{D}, \circ, \{e_0\} \rangle$, where:

- $\mathcal{D}$ is the set of all finite subsets of $\mathbb{N}$;
- $\circ$ is union of disjoint sets, undefined otherwise;
Choosing a model

Given that $F_{M,C}$ is provable, it is valid by soundness.

It’s enough to show that $M$ terminates from $C$ given only that $F_{M,C}$ is valid in some model of our choice, under some valuation of our choice.

We use the RAM-domain model $\langle D, \circ, \{e_0\} \rangle$, where:

- $D$ is the set of all finite subsets of $\mathbb{N}$;
- $\circ$ is union of disjoint sets, undefined otherwise;
- $e_0$ is the empty set.
Second main theorem

Theorem
\[ \langle L_i, n_1, n_2 \rangle \downarrow_M \text{ whenever the following sequent is valid:} \]
\[ \kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land -l_0) \vdash b \]
Second main theorem

Theorem
\langle L_i, n_1, n_2 \rangle \Downarrow_M \text{ whenever the following sequent is valid:}

\kappa(M) * l_i * p_{1}^{n_1} * p_{2}^{n_2} * (I \land -l_0) \vdash b

Proof outline. In our RAM-domain model \langle D, \circ, \{e_0\} \rangle, we have for any \rho:

\kappa(M) * l_i * p_{1}^{n_1} * p_{2}^{n_2} * (I \land -l_0) \models_{\rho} b
Second main theorem

Theorem
\(\langle L_i, n_1, n_2 \rangle \downarrow_M \) whenever the following sequent is valid:

\[
\kappa(M) * l_i * p_1^{n_1} * p_2^{n_2} * (I \land -l_0) \vdash b
\]

Proof outline. In our RAM-domain model \(\langle D, \circ, \{e_0\} \rangle\), we have for any \(\rho\):

\[
\kappa(M) * l_i * p_1^{n_1} * p_2^{n_2} * (I \land -l_0) \models_{\rho} b
\]

We want to pick \(\rho\) with \(e_0 \models_{\rho} \kappa(M)\) and \(e_0 \models_{\rho} I \land -l_0\) to get:

\[
l_i * p_1^{n_1} * p_2^{n_2} \models_{\rho} b
\]

and infer \(\langle L_i, n_1, n_2 \rangle \downarrow_M\).
\[ p^n_k \]_\rho: The (second) edge of disaster

We intend that \( l_i \ast p_1^{n_1} \ast p_2^{n_2} \) should encode configuration \( \langle L_i, n_1, n_2 \rangle \). Thus \( d \models_\rho p_k^{n_k} \) should determine the number \( n_k \).
We intend that $l_i * p_1^{n_1} * p_2^{n_2}$ should encode configuration $\langle L_i, n_1, n_2 \rangle$. Thus $d \models p_k^{n_k}$ should determine the number $n_k$.

But composition is disjoint so that, e.g., if we take $\rho(p_k) = \{h\}$ for a nonempty heap $h$, then $\rho(p_k^2) = \rho(p_k * p_k)$ is empty!
\( \llbracket p_k^n \rrbracket_\rho : The \ (second) \ edge \ of \ disaster \)

We intend that \( l_i * p_{n_1} * p_{n_2} \) should encode configuration \( \langle L_i, n_1, n_2 \rangle \). Thus \( d \models \rho \ p_{nk} \) should determine the number \( n_k \).

But composition is disjoint so that, e.g., if we take \( \rho(p_k) = \{h\} \) for a nonempty heap \( h \), then \( \rho(p_{k}^2) = \rho(p_k * p_k) \) is empty!

In general, whenever \( \rho(p_k) \) is finite we must have:

\[
\llbracket p_k^n \rrbracket_\rho = \llbracket p_k^m \rrbracket_\rho
\]

for sufficiently large \( n \) and \( m \). So we need an infinite valuation.
Choosing a valuation

We choose a valuation \( \rho \) for \( \langle \mathcal{D}, \circ, \{e_0\} \rangle \) as follows:

\[
\begin{align*}
\rho(p_1) &= \{\{2^m\} \mid m \in \mathbb{N}\} \\
\rho(p_2) &= \{\{3^m\} \mid m \in \mathbb{N}\}
\end{align*}
\]
Choosing a valuation

We choose a valuation $\rho$ for $\langle D, \circ, \{e_0\} \rangle$ as follows:

$$
\rho(p_1) = \{\{2^m\} | m \in \mathbb{N}\}
$$

$$
\rho(p_2) = \{\{3^m\} | m \in \mathbb{N}\}
$$

$$
\rho(l_i) = \{\{\delta_i^m\} | m \in \mathbb{N}\}
$$

where $\delta_i$ is a fresh prime number for each propositional variable $l_{-2}, l_{-1}, l_0, l_1, \ldots$
Choosing a valuation

We choose a valuation $\rho$ for $\langle \mathcal{D}, \circ, \{e_0\} \rangle$ as follows:

$$
\begin{align*}
\rho(p_1) &= \\{ \{2^m\} \mid m \in \mathbb{N}\} \\
\rho(p_2) &= \{\{3^m\} \mid m \in \mathbb{N}\} \\
\rho(l_i) &= \{\{\delta_i^m\} \mid m \in \mathbb{N}\}
\end{align*}
$$

where $\delta_i$ is a fresh prime number for each propositional variable $l_{-2}, l_{-1}, l_0, l_1, \ldots$

Finally, we define:

$$
\rho(b) = \bigcup_{\langle L_i, n_1, n_2 \rangle \downarrow_M} \{d \mid d \models \rho \ l_i \ast p_1^{n_1} \ast p_2^{n_2}\}
$$

so $\rho(b)$ is the set of interpretations of all terminating configurations.
Needed lemma

Lemma
For our chosen model and valuation $\rho$,

$$e_0 \models_{\rho} I \land \neg l_0.$$ 

This is easy.
Needed lemma

Lemma
For our chosen model and valuation \( \rho \),

\[ e_0 \models_\rho I \land -l_0. \]

This is easy.

Lemma

\[ e_0 \models_\rho \kappa(M). \]


**Needed lemma**

*Lemma*

*For our chosen model and valuation $\rho$,*

\[ \vdash_{\rho} I \land \neg l_0. \]

This is easy.

*Lemma*

\[ \vdash_{\rho} \kappa(M). \]

We have to show $\vdash_{\rho} \kappa(\gamma)$ for each possible instruction $\gamma$. 
Needed lemma

Lemma
For our chosen model and valuation $\rho$,

\[ e_0 \models_\rho I \land \neg l_0 . \]

This is easy.

Lemma

\[ e_0 \models_\rho \kappa(M). \]

We have to show $e_0 \models_\rho \kappa(\gamma)$ for each possible instruction $\gamma$.

This involves wrangling with the semantics of $\neg \ast$ and with the details of our valuation.
Proof of Lemma 2

If \( \kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land - l_0) \vdash b \) is valid in \( \langle \mathcal{D}, \circ, \{e_0\} \rangle \) then:

\[
\kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land - l_0) \models_{\rho} b
\]
Proof of Lemma 2

If $\kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land -l_0) \vdash b$ is valid in $\langle D, \circ, \{e_0\}\rangle$ then:

$$\kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land -l_0) \models_{\rho} b$$

Since $e_0 \models_{\rho} \kappa(M)$ we get:

$$l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land -l_0) \models_{\rho} b$$
Proof of Lemma 2

If $\kappa(M) * l_i * p_1^{n_1} * p_2^{n_2} * (I \land -l_0) \vdash b$ is valid in $\langle D, \circ, \{e_0\}\rangle$ then:

$$\kappa(M) * l_i * p_1^{n_1} * p_2^{n_2} * (I \land -l_0) \models_\rho b$$

Since $e_0 \models_\rho \kappa(M)$ we get:

$$l_i * p_1^{n_1} * p_2^{n_2} * (I \land -l_0) \models_\rho b$$

Since $e_0 \models_\rho I \land -l_0$ (because $\langle L_0, 0, 0\rangle \downarrow_M$), we get:

$$l_i * p_1^{n_1} * p_2^{n_2} \models_\rho b$$
Proof of Lemma 2

If $\kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land -l_0) \vdash b$ is valid in $\langle D, \circ, \{e_0\}\rangle$ then:

$$\kappa(M) \ast l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land -l_0) \models \rho \ b$$

Since $e_0 \models \rho \kappa(M)$ we get:

$$l_i \ast p_1^{n_1} \ast p_2^{n_2} \ast (I \land -l_0) \models \rho \ b$$

Since $e_0 \models \rho I \land -l_0$ (because $\langle L_0, 0, 0\rangle \downarrow_M$), we get:

$$l_i \ast p_1^{n_1} \ast p_2^{n_2} \models \rho \ b$$

Since $d \models \rho l_i \ast p_1^{n_1} \ast p_2^{n_2}$ uniquely determines $n_1$ and $n_2$ we conclude $\langle L_i, n_1, n_2\rangle \downarrow_M$ from definition of $\rho(b)$. 

20/ 21
Further reading

J. Brotherston and M. Kanovich.
Undecidability of propositional separation logic and its neighbours.

D. Larchey-Wendling and D. Galmiche.
Nondeterministic phase semantics and the undecidability of Boolean BI.

Computability and complexity results for a spatial assertion language for data structures.