15-819M: Data, Code, Decisions
02: Formal Modeling with Propositional Logic

André Platzer

aplatzer@cs.cmu.edu
Carnegie Mellon University, Pittsburgh, PA
Outline

1. Formal Modeling

2. Propositional Logic
   - Syntax
   - Semantics
   - Sequent Calculus
   - DPLL
   - Expressiveness

3. Temporal Logic
Outline

1. Formal Modeling

2. Propositional Logic
   - Syntax
   - Semantics
   - Sequent Calculus
   - DPLL
   - Expressiveness

3. Temporal Logic
Formalisation

Real World

Formal Model

Formalisation
Formalisation: Syntax, Semantics

Real World

Formal Languages

Interpretation

Formal Semantics

Syntax

Semantics
Formalisation: Syntax, Semantics

Real World

Syntax

Propositional Logic

Semantics

Valuation

Interpretation
Formalisation: Syntax, Semantics, Proving

Real World

Syntax

Semantics

Temporal Logic
Promela
All Runs $\sigma =$
Transition System

How to do proving?
Formal Verification: Model Checking

Real World

Syntax

Sem.

Promela

Transition System

TL
Formal Verification: Model Checking

- Real World
- Formal Model
- Formalisation
- Formal Languages
- Formal Semantics
- Syntax
- Sem.
- TL
- Promela
- Transition System
- Büchi Automaton

Translation of Negation
Formal Verification: Model Checking

- Real World
- Formal Model
- Formalisation
- Formal Languages
- Formal Semantics
- Syntax
- Semantics

Translation of Negation

Promela

Translation System

Büchi Automaton

Product accepts no run?

Transition System

Syntax

André Platzer (CMU)
Syntax, Semantics, Calculus

- Syntax
- Formula/Program
- Semantics
  - "valid"
- Calculus
  - "Derivable"
Syntax, Semantics, Calculus

Syntax

Formula/Program

Semantics

"valid"

Calculus

"Derivable"

André Platzer (CMU)
Syntax, Semantics, Calculus

Syntax
Formula/Program

Semantics
“valid”

Completeness

Calculus
“Derivable”

Soundness
Outline

1 Formal Modeling

2 Propositional Logic
   - Syntax
   - Semantics
   - Sequent Calculus
   - DPLL
   - Expressiveness

3 Temporal Logic
Propositional Logic

Propositional Formulas

Mapping
Var → \{T, F\}

Sequent Calculus

SAT Solver

André Platzer (CMU)
Propositional Logic: Syntax

Propositional Formulas

Mapping
Var → \{T, F\}

Sequent Calculus
SAT Solver
**Syntax of Propositional Logic**

<table>
<thead>
<tr>
<th>Definition (Signature)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A set of <strong>Propositional Variables</strong> $\mathcal{P}$ (with typical elements $p, q, r, \ldots$)</td>
</tr>
</tbody>
</table>
Syntax of Propositional Logic

**Definition (Signature)**

A set of **Propositional Variables** $\mathcal{P}$ (with typical elements $p, q, r, \ldots$)

**Propositional Connectives**

true false & | ! − $\rightarrow$ $\leftrightarrow$
Syntax of Propositional Logic

Definition (Signature)
A set of Propositional Variables $\mathcal{P}$ (with typical elements $p, q, r, \ldots$)

Propositional Connectives
true false & | ! → ↔

Definition (Propositional Formulas $For_0$)
- Truth constants true, false and variables $\mathcal{P}$ are formulas
- If $\phi$ and $\psi$ are formulas then
  - $!\phi, (\phi \& \psi), (\phi | \psi), (\phi \rightarrow \psi), (\phi \leftrightarrow \psi)$
  - are also formulas
- There are no other formulas (inductive definition)
Syntax of Propositional Logic

Definition (Signature)
A set of Propositional Variables $\mathcal{P}$ (with typical elements $p, q, r, \ldots$)

Propositional Connectives (KeY notation)
true false $\&$ $|$ $!$ $\rightarrow$ $\leftrightarrow$

Definition (Propositional Formulas $\text{For}_0$)
- Truth constants true, false and variables $\mathcal{P}$ are formulas
- If $\phi$ and $\psi$ are formulas then
  $$!\phi, (\phi \& \psi), (\phi | \psi), (\phi \rightarrow \psi), (\phi \leftrightarrow \psi)$$
  are also formulas
- There are no other formulas (inductive definition)
Remark on Concrete Syntax

<table>
<thead>
<tr>
<th></th>
<th>Text book</th>
<th>KeY</th>
<th>SPIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negation</td>
<td>\neg</td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td>Conjunction</td>
<td>\wedge</td>
<td>&amp;</td>
<td>&amp;&amp;</td>
</tr>
<tr>
<td>Disjunction</td>
<td>\vee</td>
<td></td>
<td>|</td>
</tr>
<tr>
<td>Implication</td>
<td>\rightarrow, \supset</td>
<td>\rightarrow</td>
<td>\rightarrow</td>
</tr>
<tr>
<td>Equivalence</td>
<td>\leftrightarrow</td>
<td>\leftrightarrow</td>
<td>\leftrightarrow</td>
</tr>
</tbody>
</table>

Today, we use KeY notation. Be flexible during the course!
Remark on Concrete Syntax

<table>
<thead>
<tr>
<th>Text book</th>
<th>KeY</th>
<th>SPIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negation</td>
<td>¬</td>
<td>!</td>
</tr>
<tr>
<td>Conjunction</td>
<td>∧</td>
<td>&amp;</td>
</tr>
<tr>
<td>Disjunction</td>
<td>∨</td>
<td></td>
</tr>
<tr>
<td>Implication</td>
<td>→, ⊃</td>
<td>→</td>
</tr>
<tr>
<td>Equivalence</td>
<td>↔</td>
<td>↔</td>
</tr>
</tbody>
</table>

Today, we use KeY notation. Be flexible during the course!
Propositional Logic: Semantics

Propositional Formulas

Mapping
Var → {T, F}

Sequent Calculus
SAT Solver
Propositional Logic: Semantics

Propositional Formulas

Mapping
Var → \{ T, F \}

Sequent Calculus

SAT Solver
Definition (Interpretation $\mathcal{I}$)

Assigns a truth value to each propositional variable

$$\mathcal{I} : \mathcal{P} \rightarrow \{T, F\}$$
### Definition (Interpretation $\mathcal{I}$)
Assigns a truth value to each propositional variable

\[ \mathcal{I} : \mathcal{P} \rightarrow \{ T, F \} \]

### Definition (Valuation function)
$val_\mathcal{I}$: Continuation of $\mathcal{I}$ on $For_0$

\[ val_\mathcal{I} : For_0 \rightarrow \{ T, F \} \]

- $val_\mathcal{I}(p_i) = \mathcal{I}(p_i)$
- $val_\mathcal{I}(\text{true}) = T$
- $val_\mathcal{I}(\text{false}) = F$

(cont’d next page)
Semantics of Propositional Logic

Definition (Valuation function . . .)

\[ \text{val}_I(! \phi) = \begin{cases} T & \text{if } \text{val}_I(\phi) = F \\ F & \text{otherwise} \end{cases} \]

\[ \text{val}_I(\phi \& \psi) = \begin{cases} T & \text{if } \text{val}_I(\phi) = T \text{ and } \text{val}_I(\psi) = T \\ F & \text{otherwise} \end{cases} \]

\[ \text{val}_I(\phi | \psi) = \begin{cases} T & \text{if } \text{val}_I(\phi) = T \text{ or } \text{val}_I(\psi) = T \\ F & \text{otherwise} \end{cases} \]

\[ \text{val}_I(\phi \rightarrow \psi) = \begin{cases} T & \text{if } \text{val}_I(\phi) = F \text{ or } \text{val}_I(\psi) = T \\ F & \text{otherwise} \end{cases} \]

\[ \text{val}_I(\phi \leftrightarrow \psi) = \begin{cases} T & \text{if } \text{val}_I(\phi) = \text{val}_I(\psi) \\ F & \text{otherwise} \end{cases} \]
Example (Formula)

\[ p \rightarrow (q \rightarrow p) \]
Example (Formula)

\[ p \rightarrow (q \rightarrow p) \]

Example (Interpretation)

One of four different ones on \( \mathcal{P} = \{p, q\} \) that are possible:

\[ I(p) = T \]
\[ I(q) = F \]
Example (Formula)

\[ p \rightarrow (q \rightarrow p) \]

Example (Interpretation)

One of four different ones on \( \mathcal{P} = \{p, q\} \) that are possible:

\[ \mathcal{I}(p) = T \]
\[ \mathcal{I}(q) = F \]

Example

\[ \text{val}_\mathcal{I}(q \rightarrow p) = \]
Example (Formula)

\[ p \rightarrow (q \rightarrow p) \]

Example (Interpretation)

One of four different ones on \( \mathcal{P} = \{p, q\} \) that are possible:

\[ I(p) = T \]
\[ I(q) = F \]

Example

\[ \text{val}_I(q \rightarrow p) = T \]
Example (Formula)

\[ p \rightarrow (q \rightarrow p) \]

Example (Interpretation)

One of four different ones on \( \mathcal{P} = \{p, q\} \) that are possible:

\[ I(p) = T \]
\[ I(q) = F \]

Example

\[ \text{val}_I( q \rightarrow p ) = T \]
\[ \text{val}_I( p \rightarrow (q \rightarrow p) ) = \]
Example (Formula)

$$p \rightarrow (q \rightarrow p)$$

Example (Interpretation)

One of four different ones on $\mathcal{P} = \{p, q\}$ that are possible:

$I(p) = T$
$I(q) = F$

Example

$$\text{val}_I(q \rightarrow p) = T$$
$$\text{val}_I(p \rightarrow (q \rightarrow p)) = T$$
Semantic Notions of Propositional Logic

Let $\phi \in \text{For}_0$, $\Gamma \subset \text{For}_0$

**Definition (Validity and Consequence Relation, overloading $\models$)**

$\phi$ is valid in $\mathcal{I}$ (write: $\mathcal{I} \models \phi$) iff $\text{val}_{\mathcal{I}}(\phi) = T$

$\phi$ follows from $\Gamma$ (write: $\Gamma \models \phi$) iff for all interpretations $\mathcal{I}$:

If $\mathcal{I} \models \psi$ for all $\psi \in \Gamma$ then also $\mathcal{I} \models \phi$

**Definition (Satisfiability, Validity)**

A formula is satisfiable if it is valid in some interpretation. If $\phi$ is valid in every interpretation, i.e. $\emptyset \models \phi$ (short: $\models \phi$) then $\phi$ is called logically valid.
Semantic Notions of Propositional Logic

Let $\phi \in For_0$, $\Gamma \subset For_0$

Definition (Validity and Consequence Relation, overloading $\models$)

$\phi$ is valid in $\mathcal{I}$ (write: $\mathcal{I} \models \phi$) iff $val_\mathcal{I}(\phi) = T$

$\phi$ follows from $\Gamma$ (write: $\Gamma \models \phi$) iff for all interpretations $\mathcal{I}$:

If $\mathcal{I} \models \psi$ for all $\psi \in \Gamma$ then also $\mathcal{I} \models \phi$

Definition (Satisfiability, Validity)

A formula is **satisfiable** if it is valid in **some** interpretation.

If $\phi$ is valid in **every** interpretation, i.e.

$$\emptyset \models \phi \quad \text{(short: } \models \phi)$$

then $\phi$ is called **logically valid**.
Example (Formula)

\[ p \rightarrow (q \rightarrow p) \]
Example (Formula)

\[ p \rightarrow (q \rightarrow p) \]

Is this formula valid?

\[ \models p \rightarrow (q \rightarrow p) \] ?
Examples

\[ p \land \left( (\neg p) \lor q \right) \]

Satisfiable?

I(\neg p) = T, I(q) = T

Therefore, also not valid!

\[ p \land \left( (\neg p) \lor q \right) \lor r = q \lor r \]

Does it hold?

Yes. Why?

André Platzer (CMU)
Examples

$p \land (\neg p \lor q)$

Satisfiable?  

Yes. Why?

Therefore, also not valid!

$p \land (\neg p \lor q) \lor r = q \lor r$
Examples

$p \land ((\neg p) \lor q)$

Satisfiable?  

Yes. Why?

Satisfying Interpretation?

$I(p) = T, I(q) = T$
Examples

\( p \& ((!p) \mid q) \)

Satisfiable?  
\( \checkmark \)

Satisfying Interpretation?  
\( \mathcal{I}(p) = T, \mathcal{I}(q) = T \)

Therefore, also not valid!
Examples

\[ p \land ((\neg p) \lor q) \]

Satisfiable?  
Yes.

Satisfying Interpretation?  
\[ I(p) = T, I(q) = T \]

Other Satisfying Interpretations?
Examples

$p \& ((\neg p) \mid q)$

Satisfiable?  \(\checkmark\)
Satisfying Interpretation?  \(\mathcal{I}(p) = T, \mathcal{I}(q) = T\)
Other Satisfying Interpretations?  \(\times\)
Examples

\[ p \land ((\neg p) \lor q) \]

Satisfiable? \(\checkmark\)
Satisfying Interpretation? \(\mathcal{I}(p) = T, \mathcal{I}(q) = T\)
Other Satisfying Interpretations? \(\times\)
Therefore, also not valid!
Examples

\[ p \land (\neg p \lor q) \]

Satisfiable? \( \checkmark \)
Satisfying Interpretation? \( \mathcal{I}(p) = T, \mathcal{I}(q) = T \)
Other Satisfying Interpretations? \( \times \)
Therefore, also not valid!

\[ p \land (\neg p \lor q) \models q \lor r \]

Does it hold?
Examples

\[ p \land ((\neg p) \lor q) \]

Satisfiable? \( \checkmark \)
Satisfying Interpretation? \( I(p) = T, I(q) = T \)
Other Satisfying Interpretations? \( \times \)
Therefore, also not valid!

\[ p \land ((\neg p) \lor q) \models q \lor r \]

Does it hold? \( \text{Yes.} \)
Why?
Propositional Logic: Calculus

Propositional Formulas

|$\vdash$|

Mapping

Var → \{T, F\}

|$|=|

Sequent Calculus
Propositional Logic: Calculus

Propositional Formulas

Mapping
Var → \{ T, F \}

Sequent Calculus
Establish $\models \phi$ by finite, syntactic transformation of $\phi$
Reasoning by Syntactic Transformation

Establish $\models \phi$ by finite, syntactic transformation of $\phi$

Definition ((Logic) Calculus)

A set of (decidable) syntactic transformation rules $\mathcal{R}$ defining a relation $\vdash \subseteq For_0$ such that

- $\vdash \phi$ implies $\models \phi$: Soundness (required)
- $\models \phi$ implies $\vdash \phi$: Completeness (desirable)
Reasoning by Syntactic Transformation

Establish $\models \phi$ by finite, syntactic transformation of $\phi$

Definition ((Logic) Calculus)

A set of (decidable) syntactic transformation rules $\mathcal{R}$ defining a relation $\vdash \subseteq \text{For}_0$ such that

- $\vdash \phi$ implies $\models \phi$: Soundness (required)
- $\models \phi$ implies $\vdash \phi$: Completeness (desirable)

Sequent Calculus based on notion of sequent

$$\underbrace{\psi_1, \ldots, \psi_m}_\text{Antecedent} \implies \underbrace{\phi_1, \ldots, \phi_n}_\text{Succedent}$$

has same semantics as

$$\left( \psi_1 \land \cdots \land \psi_m \right) \rightarrow \left( \phi_1 \mid \cdots \mid \phi_n \right)$$

$$\{\psi_1, \ldots, \psi_m\} \models \phi_1 \mid \cdots \mid \phi_n$$
ψ₁, ..., ψₘ  ⇒  φ₁, ..., φₙ

Consider antecedent/succedent as sets of formulas, possibly empty
**Notation for Sequents**

\[
\psi_1, \ldots, \psi_m \implies \phi_1, \ldots, \phi_n
\]

Consider antecedent/succedent as sets of formulas, possibly empty

**Definition (Schema Variables)**

\( \phi, \psi, \ldots \) match formulas, \( \Gamma, \Delta, \ldots \) match sets of formulas

Characterize infinitely many sequents with a single schematic sequent

\[
\Gamma \implies \Delta, \phi \& \psi
\]

Matches any sequent with occurrence of conjunction in succedent

Call \( \phi \& \psi \) main formula and \( \Gamma, \Delta \) side formulas of sequent

Any sequent of the form \( \Gamma, \phi \implies \Delta, \phi \) is logically valid: axiom
Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible

**RuleName**

Premisses

\[
\frac{\Gamma_1 \implies \Delta_1 \quad \cdots \quad \Gamma_r \implies \Delta_r}{\Gamma \implies \Delta}
\]

Conclusion

Example

**andRight**

\[
\Gamma = \implies \phi, \quad \Delta \Gamma = \implies \psi, \quad \Delta \Gamma = \implies \phi \land \psi
\]

Sound rule (essential):

\[
\mid = (\Gamma_1 \implies \Delta_1 \land \cdots \land \Gamma_r \implies \Delta_r) \quad - > \quad (\Gamma = \implies \Delta)
\]

Complete rule (desirable):

\[
\mid = (\Gamma = \implies \Delta) \quad - > \quad (\Gamma_1 \implies \Delta_1 \land \cdots \land \Gamma_r \implies \Delta_r)
\]

Admissible to have no premisses (iff conclusion is valid, eg axiom)
Sequent Calculus Rules of Propositional Logic

Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible

**RuleName**

\[ \frac{\Gamma_1 \Rightarrow \Delta_1 \quad \cdots \quad \Gamma_r \Rightarrow \Delta_r}{\Gamma \Rightarrow \Delta} \]

**Premisses**

**Conclusion**

**Example**

**andRight**

\[ \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \ \& \ \psi, \Delta} \]
Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible

\[ \Gamma_1 \Rightarrow \Delta_1 \quad \cdots \quad \Gamma_r \Rightarrow \Delta_r \]

\[ \Gamma \Rightarrow \Delta \]

\textbf{Example}

\textbf{andRight}

\[ \Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta \]

\[ \Gamma \Rightarrow \phi \& \psi, \Delta \]

\textbf{Sound} rule (essential): \[ \models (\Gamma_1 \Rightarrow \Delta_1 \& \cdots \& \Gamma_r \Rightarrow \Delta_r) \rightarrow (\Gamma \Rightarrow \Delta) \]
Sequent Calculus Rules of Propositional Logic

Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible.

\[
\text{RuleName} \quad \begin{array}{c}
\Gamma_1 \Rightarrow \Delta_1 \\
\cdots \\
\Gamma_r \Rightarrow \Delta_r \\
\hline
\Gamma \Rightarrow \Delta
\end{array}
\]

Example

\[
\text{andRight} \quad \begin{array}{c}
\Gamma \Rightarrow \phi, \Delta \\
\Gamma \Rightarrow \psi, \Delta
\end{array} \quad \Gamma \Rightarrow \phi \ & \ & \psi, \Delta
\]

Sound rule (essential):
\[
\models (\Gamma_1 \Rightarrow \Delta_1 \& \cdots \& \Gamma_r \Rightarrow \Delta_r) \rightarrow (\Gamma \Rightarrow \Delta)
\]

Complete rule (desirable):
\[
\models (\Gamma \Rightarrow \Delta) \rightarrow (\Gamma_1 \Rightarrow \Delta_1 \& \cdots \& \Gamma_r \Rightarrow \Delta_r)
\]

Admissible to have no premisses (iff conclusion is valid, e.g. axiom)
### Rules of Propositional Sequent Calculus

<table>
<thead>
<tr>
<th>main</th>
<th>left side (antecedent)</th>
<th>right side (succedent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>( \Gamma \implies \phi, \Delta )</td>
<td>( \Gamma, \phi \implies \Delta )</td>
</tr>
<tr>
<td></td>
<td>( \Gamma, !\phi \implies \Delta )</td>
<td>( \Gamma \implies !\phi, \Delta )</td>
</tr>
</tbody>
</table>
Rules of Propositional Sequent Calculus

<table>
<thead>
<tr>
<th>main</th>
<th>left side (antecedent)</th>
<th>right side (succedent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>$\Gamma \Rightarrow \phi, \Delta$</td>
<td>$\Gamma, \phi \Rightarrow \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, !\phi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow !\phi, \Delta$</td>
</tr>
<tr>
<td>and</td>
<td>$\Gamma, \phi, \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi, \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma \Rightarrow \psi, \Delta$</td>
<td>$\Gamma \Rightarrow \phi &amp; \psi, \Delta$</td>
</tr>
<tr>
<td>main</td>
<td>left side (antecedent)</td>
<td>right side (succedent)</td>
</tr>
<tr>
<td>------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>not</td>
<td>$\Gamma \Rightarrow \phi, \Delta$</td>
<td>$\Gamma, \phi \Rightarrow \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, !\phi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow !\phi, \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, \phi, \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, \phi &amp; \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi &amp; \psi, \Delta$</td>
</tr>
<tr>
<td>and</td>
<td>$\Gamma, \phi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi, \psi, \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, \phi \mid \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi \mid \psi, \Delta$</td>
</tr>
<tr>
<td>or</td>
<td>$\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta$</td>
<td>$\Gamma \Rightarrow \phi &amp; \psi, \Delta$</td>
</tr>
</tbody>
</table>
## Rules of Propositional Sequent Calculus

<table>
<thead>
<tr>
<th>main</th>
<th>left side (antecedent)</th>
<th>right side (succedent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>[ \Gamma \Rightarrow \phi, \Delta ]</td>
<td>[ \Gamma, \phi \Rightarrow \Delta ]</td>
</tr>
<tr>
<td></td>
<td>[ \Gamma, ! \phi \Rightarrow \Delta ]</td>
<td>[ \Gamma \Rightarrow ! \phi, \Delta ]</td>
</tr>
<tr>
<td>and</td>
<td>[ \Gamma, \phi, \psi \Rightarrow \Delta ]</td>
<td>[ \Gamma \Rightarrow \phi, \Delta ]  [ \Gamma \Rightarrow \psi, \Delta ]  [ \Gamma \Rightarrow \phi &amp; \psi, \Delta ]</td>
</tr>
<tr>
<td></td>
<td>[ \Gamma, \phi &amp; \psi \Rightarrow \Delta ]</td>
<td>[ \Gamma \Rightarrow \phi &amp; \psi, \Delta ]</td>
</tr>
<tr>
<td>or</td>
<td>[ \Gamma, \phi \Rightarrow \Delta ]  [ \Gamma, \psi \Rightarrow \Delta ]</td>
<td>[ \Gamma \Rightarrow \phi \Rightarrow \psi, \Delta ]</td>
</tr>
<tr>
<td></td>
<td>[ \Gamma, \phi \mid \psi \Rightarrow \Delta ]</td>
<td>[ \Gamma \Rightarrow \phi \mid \psi, \Delta ]</td>
</tr>
<tr>
<td>imp</td>
<td>[ \Gamma \Rightarrow \phi, \Delta ]  [ \Gamma, \psi \Rightarrow \Delta ]</td>
<td>[ \Gamma, \phi \Rightarrow \psi, \Delta ]</td>
</tr>
<tr>
<td></td>
<td>[ \Gamma \Rightarrow \phi \rightarrow \psi \Rightarrow \Delta ]</td>
<td>[ \Gamma \Rightarrow \phi \rightarrow \psi, \Delta ]</td>
</tr>
</tbody>
</table>
### Rules of Propositional Sequent Calculus

<table>
<thead>
<tr>
<th>Rule</th>
<th>Left Side (Antecedent)</th>
<th>Right Side (Succedent)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>not</strong></td>
<td>$\Gamma \Rightarrow \phi, \Delta$</td>
<td>$\Gamma, \phi \Rightarrow \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, ! \phi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow ! \phi, \Delta$</td>
</tr>
<tr>
<td><strong>and</strong></td>
<td>$\Gamma, \phi, \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi, \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, \phi &amp; \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \psi, \Delta$</td>
</tr>
<tr>
<td><strong>or</strong></td>
<td>$\Gamma, \phi \Rightarrow \Delta$, $\Gamma, \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi, \psi, \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, \phi \mid \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi \mid \psi, \Delta$</td>
</tr>
<tr>
<td><strong>imp</strong></td>
<td>$\Gamma \Rightarrow \phi, \Delta$, $\Gamma, \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi \Rightarrow \psi, \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, \phi \Rightarrow \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi \Rightarrow \psi, \Delta$</td>
</tr>
<tr>
<td><strong>close</strong></td>
<td>$\Gamma, \phi \Rightarrow \phi, \Delta$</td>
<td>$\Gamma \Rightarrow \text{true}, \Delta$</td>
</tr>
<tr>
<td><strong>true</strong></td>
<td>$\Gamma \Rightarrow \text{true}, \Delta$</td>
<td>$\Gamma, \text{false} \Rightarrow \Delta$</td>
</tr>
<tr>
<td><strong>false</strong></td>
<td>$\Gamma, \text{false} \Rightarrow \Delta$</td>
<td></td>
</tr>
</tbody>
</table>
Justify rules by applying semantic definitions
Justify rules by applying semantic definitions

\[ \text{orRight} \quad \frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \mid \psi, \Delta} \]

Follows directly from semantics of sequents
Justification of Rules

Justify rules by applying semantic definitions

\[ \text{orRight} \quad \frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \mid \psi, \Delta} \]

Follows directly from semantics of sequents

\[ \text{andRight} \quad \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \& \psi, \Delta} \]

\[ \Gamma \rightarrow (\phi \& \psi) \mid \Delta \quad \text{iff} \quad \Gamma \rightarrow \phi \mid \Delta \quad \text{and} \quad \Gamma \rightarrow \psi \mid \Delta \]

Distributivity of & over | and \rightarrow
Sequent Calculus Proofs

**Goal** to prove: $G \equiv \psi_1, \ldots, \psi_m \Rightarrow \phi_1, \ldots, \phi_n$

- find rule $R$ whose conclusion matches $G$
- instantiate $R$ such that conclusion identical to $G$
- recursively find proofs for resulting premisses $G_1, \ldots, G_r$
- tree structure with goal as root
- close proof branch when rule without premiss encountered

Goal-directed proof search

In KeY tool proof displayed as a tree
A Simple Proof

\[ p \Rightarrow q, \quad q \Rightarrow (p \& (p \rightarrow q)) \Rightarrow q \]
A Simple Proof

\[
\begin{align*}
p & \Rightarrow q, \\
p & = q, \\
\end{align*}
\]

\[
p \land (p \rightarrow q) \Rightarrow q \\
\Rightarrow (p \land (p \rightarrow q)) \rightarrow q
\]
A Simple Proof

\[ p, (p \rightarrow q) \Rightarrow q \]
\[ p \land (p \rightarrow q) \Rightarrow q \]
\[ \Rightarrow (p \land (p \rightarrow q)) \rightarrow q \]
A Simple Proof

\[ p \implies q, \quad p \implies q \]

\[ p, \ (p \rightarrow q) \implies q \]

\[ p \land (p \rightarrow q) \implies q \]

\[ \implies (p \land (p \rightarrow q)) \rightarrow q \]
A Simple Proof

\[
\begin{align*}
\text{\textsc{close}} & \quad \ast \\
\quad & \quad \quad \left( p \Rightarrow q, \ p \right) \\
\end{align*}
\]

\[
\begin{align*}
\text{\textsc{close}} & \quad \ast \\
\quad & \quad \quad \left( p, q \Rightarrow q \right) \\
\quad & \quad \quad \left( p, (p \rightarrow q) \Rightarrow q \right) \\
\quad & \quad \quad \left( p \land (p \rightarrow q) \Rightarrow q \right) \\
\end{align*}
\]

\[
\Rightarrow (p \land (p \rightarrow q)) \rightarrow q
\]
A Simple Proof

\[
\begin{align*}
\text{CLOSE} & \quad \ast \\
p & \implies q, p & \ast & \implies q \\
p, q & \implies q \\
p, (p \implies q) & \implies q \\
p \land (p \implies q) & \implies q \\
\implies (p \land (p \implies q)) & \implies q
\end{align*}
\]

A proof is \textit{closed} iff all its branches are closed

Demo

Examples/prop.key
refute(S):
    while false $\not\in$ S do
        if S = $\emptyset$ then return sat
        if S does not contain unit clause then
            P := choose variable
            /* split on P */
            refute(S with P:=false);
            refute(S with P:=true);
        else
            K := choose unit clause from S
            /* propagate K */
            drop all clauses containing K
            drop complement of K from all clauses
        end if
    end while
return unsat
\( A \mid B \mid C \)
\( \neg A \mid B \mid \neg D \)
\( \neg A \mid C \)
\( \neg A \mid \neg C \mid D \)
\( A \mid \neg C \)
\( \neg B \)
\begin{center}
\begin{align*}
A & \mid B & \mid C \\
\neg A & \mid B & \mid \neg D \\
\neg A & \mid C \\
\neg A & \mid \neg C & \mid D \\
A & \mid \neg C \\
\neg B
\end{align*}
\end{center}
\[
\begin{align*}
A & | C \\
!A & | !D \\
!A & | C \\
!A & | !C | D \\
A & | !C \\
&
A & | C \\
!A & | !D \\
!A & | C \\
!A & | !C | D \\
A & | !C \\
\end{align*}
\]
refute (with $A:=\text{true}$)

$A \mid C$
$\neg A \mid \neg D$
$\neg A \mid C$
$\neg A \mid \neg C \mid D$
$A \mid \neg C$

refute (with $A:=\text{false}$)

$A \mid C$
$\neg A \mid \neg D$
$\neg A \mid C$
$\neg A \mid \neg C \mid D$
$A \mid \neg C$
refute(with A:=true)

\[ A \mid C \]
\[ \neg A \mid \neg D \]
\[ \neg A \mid C \]
\[ \neg A \mid C \mid D \]
\[ A \mid \neg C \]

refute(with A:=false)

\[ A \mid C \]
\[ \neg A \mid \neg D \]
\[ \neg A \mid C \]
\[ \neg A \mid C \mid D \]
\[ A \mid \neg C \]
A | C
!A | !D
!A | C
!A | !C | D
A | !C

propagate(!B)

A | C
!A | !D
!A | C
!A | !C | D
A | !C

refute(with A:=true)

!D
C
!C | D

propagate(C)

refute(with A:=false)

A | C
!A | !D
!A | C
!A | !C | D
A | !C
refute (with $A:=\text{true}$)

\[
\begin{align*}
!D \\
C \\
!C | D
\end{align*}
\]

propagate ($C$)

\[
\begin{align*}
C \\
!C | D
\end{align*}
\]

refute (with $A:=\text{false}$)

\[
\begin{align*}
A | C \\
!A | !D \\
!A | C \\
!A | !C | D \\
A | !C
\end{align*}
\]
\begin{align*}
A & \mid C \\
\neg A & \mid \neg D \\
\neg A & \mid C \\
\neg A & \mid C \mid D \\
A & \mid \neg C \\
\neg A & \mid \neg C \mid D \\
A & \mid \neg C \\
\neg A & \mid \neg C \mid D \\
A & \mid C
\end{align*}

\textbf{propagate(! B)}

\begin{align*}
A & \mid C \\
\neg A & \mid \neg D \\
\neg A & \mid C \\
\neg A & \mid C \mid D \\
A & \mid \neg C \\
\neg A & \mid \neg C \mid D \\
A & \mid \neg C
\end{align*}

\textbf{propagate(C)}

\begin{align*}
\neg D \\
C \\
\neg C \mid D \\
D
\end{align*}

\textbf{propagate(! D)}

\begin{align*}
\neg D \\
D
\end{align*}

\textbf{refute(with A:=true)}

\begin{align*}
\neg D \\
\neg C \\
\neg C \mid D \\
D
\end{align*}

\textbf{refute(with A:=false)}

\begin{align*}
A & \mid C \\
\neg A & \mid \neg D \\
\neg A & \mid C \\
\neg A & \mid C \mid D \\
A & \mid \neg C \\
\neg A & \mid \neg C \mid D \\
A & \mid \neg C
\end{align*}
\[
\begin{align*}
A & \mid C \\
\neg A & \mid \neg D \\
\neg A & \mid C \\
\neg A & \mid C \mid D \quad \text{propagate}(\neg B) \\
\neg A & \mid \neg C \mid D \\
A & \mid \neg C \\
\end{align*}
\]

refute\( (\text{with } A:=\text{true}) \)

\[
\begin{align*}
\neg D \\
C & \quad \text{propagate}(C) \\
\neg C & \mid D \\
\end{align*}
\]

unsat! empty clause

refute\( (\text{with } A:=\text{false}) \)

\[
\begin{align*}
A & \mid C \\
\neg A & \mid \neg D \\
\neg A & \mid C \\
\neg A & \mid \neg C \mid D \\
\neg A & \mid \neg C \\
\end{align*}
\]
propagate(! B)

refute(with A:=true)

propagate( C)
propagate(! D)

unsat! empty clause

refute(with A:=false)

André Platzer (CMU)
refute (with A:=true)


refute (with A:=false)

André Platzer (CMU)
refute(with $A := \text{true}$)

\[
\begin{align*}
&! A \\
&! A \\
&! A \\
&! A \\
&! A \\
&A \\
\end{align*}
\]

propagate($! B$)

\[
\begin{align*}
&! A \\
&! A \\
&! A \\
&! A \\
&! A \\
&A \\
\end{align*}
\]

propagate($! C$)

\[
\begin{align*}
&! D \\
&C \\
&! C \\
&! C \\
&A \\
\end{align*}
\]

unsat! empty clause

refute(with $A := \text{false}$)

\[
\begin{align*}
&! A \\
&! A \\
&! A \\
&! A \\
&! A \\
&A \\
\end{align*}
\]

propagate($C$)

\[
\begin{align*}
&! D \\
&C \\
&! C \\
&! C \\
&A \\
\end{align*}
\]

unsat! empty clause

! C

propagate($C$)

unsat! empty clause

! C
Finite set of elements \( N = \{1, \ldots, n\} \)

Let \( p_{ij} \) denote \( p(i) = j \). \( p \) is a permutation on \( N \) is expressible . . .

Groups, Latin squares, Sudoku, . . .

Even finite numbers (e.g., bitwise encoding)
Limitations of Propositional Logic

**Fixed, finite number of objects**
- Cannot express: let \( g \) be group with *arbitrary* number of elements

**No functions or relations with arguments**
- Can express: finite function/relation table \( p_{ij} \)
- Cannot express: properties of function/relation on all arguments, e.g., \(+\) is associative

**Static interpretation**
- Programs change value of their variables, e.g., via assignment, call, etc.
- Propositional formulas look at one *single* interpretation at a time
Beyond the Limitations of Propositional Logic

- Propositional Logic
- First-order Logic
- Temporal Logic

+ state change
+ functions
Beyond the Limitations of Propositional Logic

- Propositional Logic
- First-order Logic
- Dynamic Logic
- Temporal Logic

+ state change + functions + state change + functions + functions
Beyond the Limitations of Propositional Logic

Propositional Logic

First-order Logic

Dynamic Logic

Temporal Logic

+state change +functions

+functions

SPIN

André Platzer (CMU)

15-819M/02: Data, Code, Decisions
Beyond the Limitations of Propositional Logic

- Propositional Logic
- First-order Logic
- Temporal Logic
- Dynamic Logic

+ state change + functions
+ functions

SPIN
KeY
Outline

1. Formal Modeling

2. Propositional Logic
   - Syntax
   - Semantics
   - Sequent Calculus
   - DPLL
   - Expressiveness

3. Temporal Logic
Transition Systems / Kripke Structures

\[ s_a \rightarrow s_b \rightarrow s_c \rightarrow s_d \]

\[ p = T; \quad q = p; \quad p = F; \]

Notation

- name
- interp.

update
Each state has its own propositional interpretation!
Computations, or *runs*, are infinite paths through states
Infinitely many different runs
How to express (for example) that either $p$ or $q$ changes its value infinitely often in each run?
Linear Temporal Logic

An extension of propositional logic that allows to specify properties of sets of runs
Linear Temporal Logic: Syntax

An extension of propositional logic that allows to specify properties of sets of runs

Syntax

Based on propositional signature and syntax. Extension with three connectives:

- **Always** If \( \phi \) is a formula then so is \([\square \phi]\)
- **Sometimes** If \( \phi \) is a formula then so is \(<>\phi\)
- **Until** If \( \phi \) and \( \psi \) are formulas then so is \(\phi \mathcal{U} \psi\)

Concrete Syntax

<table>
<thead>
<tr>
<th></th>
<th>text book</th>
<th>SPIN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Always</strong></td>
<td>(\square)</td>
<td>([\square])</td>
</tr>
<tr>
<td><strong>Sometimes</strong></td>
<td>(\diamond)</td>
<td>(&lt;&gt;)</td>
</tr>
<tr>
<td><strong>Until</strong></td>
<td>(\mathcal{U})</td>
<td>(\mathcal{U})</td>
</tr>
</tbody>
</table>
A run $\sigma$ is an infinite chain of states

\[ s_0 \xrightarrow{I_0} s_1 \xrightarrow{I_1} s_2 \xrightarrow{I_2} s_3 \xrightarrow{I_3} s_4 \xrightarrow{I_4} \ldots \]

$I_j$ propositional interpretation of variables in $j$-th state
Write more compactly $s_0 \ s_1 \ s_2 \ s_3 \ldots$
A run $\sigma$ is an infinite chain of states

$\mathcal{I}_j$ propositional interpretation of variables in $j$-th state

Write more compactly $s_0 s_1 s_2 s_3 \ldots$

If $\sigma = s_0 s_1 \ldots$, then $\sigma|_i$ denotes the suffix $s_i s_{i+1} \ldots$ of $\sigma$. 
Semantics of Temporal Logic

Definition (Validity Relation)

Validity of temporal formula depends on runs $\sigma = s_0 s_1 \ldots$ for which the formula may, or may not, hold:

$\sigma \models p$ iff $I_0(p) = T$, for $p \in \mathcal{P}$. 

$p = T; \quad q = p; \quad p = F; \quad q = F; \quad p = T;$

$\begin{array}{c}
s_a \quad F F \quad s_b \quad T F \quad s_c \quad T T \quad s_d \quad F T \\
p = T; \quad q = p; \quad p = F; \quad q = F; \quad p = T;
\end{array}$
Definition (Validity Relation)

Validity of temporal formula depends on runs $\sigma = s_0 s_1 \ldots$ for which the formula may, or may not, hold:

$\sigma \models p$ \iff $I_0(p) = T$, for $p \in \mathcal{P}$.
Semantics of Temporal Logic

Definition (Validity Relation)

Validity of temporal formula depends on runs $\sigma = s_0 s_1 \ldots$ for which the formula may, or may not, hold:

$\sigma \models p$ iff $I_0(p) = T$, for $p \in P$.

$\sigma \models !\phi$ iff not $\sigma \models \phi$ (write $\sigma \not\models \phi$)
**Semantics of Temporal Logic**

![State transition diagram]

**Definition (Validity Relation)**

Validity of temporal formula depends on runs $\sigma = s_0 \ s_1 \ldots$ for which the formula may, or may not, hold:

- $\sigma \models p$ iff $I_0(p) = T$, for $p \in P$.
- $\sigma \models ! \phi$ iff not $\sigma \models \phi$ (write $\sigma \not\models \phi$)
- $\sigma \models \phi \& \psi$ iff $\sigma \models \phi$ and $\sigma \models \psi$
Definition (Validity Relation)

Validity of temporal formula depends on runs \( \sigma = s_0 s_1 \ldots \) for which the formula may, or may not, hold:

- \( \sigma \models p \) \iff \( \mathcal{I}_0(p) = T \), for \( p \in \mathcal{P} \).
- \( \sigma \models ! \phi \) \iff not \( \sigma \models \phi \) (write \( \sigma \notmodels \phi \)).
- \( \sigma \models \phi \land \psi \) \iff \( \sigma \models \phi \) and \( \sigma \models \psi \).
- \( \sigma \models \phi \lor \psi \) \iff \( \sigma \models \phi \) or \( \sigma \models \psi \).
- \( \sigma \models \phi \rightarrow \psi \) \iff \( \sigma \notmodels \phi \) or \( \sigma \models \psi \).
Semantics of Temporal Logic

Definition (Validity Relation for Temporal Connectives)

Given a run $\sigma = s_0, s_1, s_2 \ldots$
Definition (Validity Relation for Temporal Connectives)

Given a run $\sigma = s_0 s_1 s_2 \ldots$

$\sigma \models [\square] \phi$ iff $\sigma|_k \models \phi$ for all $k \geq 0$
Semantics of Temporal Logic

Definition (Validity Relation for Temporal Connectives)

Given a run $\sigma = s_0 s_1 s_2 \ldots$

$\sigma \models [] \phi$ iff $\sigma|_k \models \phi$ for all $k \geq 0$

$\sigma \models <> \phi$ iff $\sigma|_k \models \phi$ for some $k \geq 0$
Semantics of Temporal Logic

Definition (Validity Relation for Temporal Connectives)

Given a run $\sigma = s_0, s_1, s_2 \ldots$

$\sigma \models []\phi$ iff $\sigma|_k \models \phi$ for all $k \geq 0$

$\sigma \models <>\phi$ iff $\sigma|_k \models \phi$ for some $k \geq 0$

$\sigma \models \phi U \psi$ iff $\sigma|_k \models \psi$ for some $k \geq 0$, and $\sigma|_{j} \models \phi$ for all $0 \leq j < k$
## Safety and Liveness Properties

### Safety Properties

Always-formulas called **safety property**: something bad never happens

Let \( \text{mutex} \) be variable that is true when two process do not access a critical resource at the same time

\[ \text{mutex} \]

\( \text{mutex} \) expresses that simultaneous access never happens
Safety and Liveness Properties

Safety Properties
Always-formulas called safety property: something bad never happens
Let $\text{mutex}$ be variable that is true when two processes do not access a critical resource at the same time
$[]\text{mutex}$ expresses that simultaneous access never happens

Liveness Properties
Sometimes-formulas called liveness property: something good happens eventually
Let $s$ be variable that is true when a process delivers a service
$<> s$ expresses that service is eventually provided
What does this mean?

\[[]<>\phi\]
Infinitely Often

\[ [\diamondsuit] \]

During a run the formulas $\phi$ will become true infinitely often.
Infinitely Often

\[
[] <> \phi
\]

During a run the formulas \( \phi \) will become true infinitely often.

What does this mean?

\[
<> [\] \phi
\]
Complex Properties

**Infinitely Often**

\[ []<>\phi \]

During a run the formulas \( \phi \) will become true infinitely often.

**Finally Always**

\[ <>[\[]\phi \]

During a run the formulas \( \phi \) will become eventually stay true indefinitely.
Definition (Validity)

\( \phi \) is valid, write \( \models \phi \), iff \( \phi \) is valid in all runs \( \sigma = s_0 s_1 \ldots \).

Recall that each run \( s_0 s_1 \ldots \) essentially is an infinite sequence of interpretations \( I_0 I_1 \ldots \).
Examples


Valid?

$<>[\neg \phi]$

Both are valid!

$[\neg]$ is reflexive and $<>$ are dual connectives
Valid?

No, there is a run in where it is not valid:
Valid?

No, there is a run in where it is not valid:

(! φ, ! φ, ! φ, ... )
Valid?

No, there is a run in where it is not valid:

\((! \phi, ! \phi, ! \phi, \ldots)\)

Valid in some run?

\(<> [\phi] \)
Examples

Valid?
No, there is a run in where it is not valid:
(!φ, !φ, !φ, ...)

Valid in some run?
Yes: (φ, φ, φ, ...)

<>[ ]φ

Both are valid!

[ ] is reflexive
and <> are dual connectives
Examples

<>[\([]\phi]

Valid?
No, there is a run in where it is not valid:
(\!(\phi, \!(\phi, \!(\phi, \ldots))

Valid in some run?
Yes: (\phi, \phi, \phi, \ldots)

\[\[]\phi \rightarrow \phi \quad (\!(\[]\phi) \leftrightarrow (<>\!\phi)

Both are valid!
Examples

\[<\!>[]{\phi}\]

Valid?
No, there is a run in where it is not valid:
\((\neg \phi, \neg \phi, \neg \phi, \ldots)\)

Valid in some run?
Yes: \((\phi, \phi, \phi, \ldots)\)

\[[]\phi \rightarrow \phi\quad (\neg []\phi) \leftrightarrow (<\!> ! \phi)\]

Both are valid!

- \([]\) is reflexive
- \([]\) and \(<\!>\) are dual connectives
Transition systems revisited

Definition (Transition System)

A Transition System $\mathcal{T} = (S, \text{Ini}, \delta, \mathcal{I})$ is given by a set of states $S$, a non-empty subset $\text{Ini} \subseteq S$ of initial states, and a transition relation $\delta \subseteq S \times S$, and $\mathcal{I}$ labeling each state $s \in S$ with a propositional interpretation $\mathcal{I}_s$.

Definition (Runs of Transition System)

A run of $\mathcal{T}$ is a run $\sigma = s_0 s_1 \ldots$, with $s_i \in S$, such that $s_0 \in \text{Ini}$ and $(s_i, s_{i+1}) \in \delta$ for all $i$. 
Validity of temporal formula is extended to transition systems in the following way:

**Definition (Validity Relation)**

Given a transition systems $\mathcal{T} = (S, Ini, \delta, \mathcal{I})$, a temporal formula $\phi$ is valid in $\mathcal{T}$ (write $\mathcal{T} \models \phi$) iff $\sigma \models \phi$ for all runs $\sigma$ of $\mathcal{T}$. 

André Platzer (CMU)

Ben-Ari  Mordechai Ben-Ari: *Principles of the Spin Model Checker*, Springer, 2008(!). Section 5.2.1 (*PROMELA* examples briefly)