

Formal Methods for Software Development

Temporal Model Checking (part 2) + First-Order Logic

Wolfgang Ahrendt

24th September 2020

Part I

Finishing Temporal Model Checking

Model Checking

Check whether a formula is valid in all runs of a transition system.

Given a transition system $\mathcal{T}$ (e.g., derived from a PROMELA program).

Verification task: is the LTL formula ϕ satisfied in all traces of $\mathcal{T}$, i.e.,

$$\mathcal{T} \models \phi \quad ?$$

LTL Model Checking—Overview

$$\mathcal{T} \models \phi \quad ?$$

1. Construct **generalised Büchi automaton** $\mathcal{GB}_{\neg\phi}$ for **negation** of ϕ
2. Construct an equivalent **normal Büchi automaton** $\mathcal{B}_{\neg\phi}$, i.e.,

$$\mathcal{L}^\omega(\mathcal{B}_{\neg\phi}) = \mathcal{L}^\omega(\mathcal{GB}_{\neg\phi})$$

3. Construct **product** $\mathcal{T} \otimes \mathcal{B}_{\neg\phi}$ (model checking graph)
4. Analyse whether $\mathcal{T} \otimes \mathcal{B}_{\neg\phi}$ has a
path π looping through an ‘accepting node’
5. If such a π is found, then

$$\mathcal{T} \not\models \phi$$

and

σ_π is a counter example.

If no such π is found, then

$$\mathcal{T} \models \phi$$

What Remains?

last lecture

- 3.–5. product of transition system and Büchi automaton (construction and analysis)

What Remains?

last lecture

- 3.–5. product of transition system and Büchi automaton (construction and analysis)

this lecture

- 2. generalised Büchi automata and their normalisation
- 1. translating LTL into generalised Büchi automata

Generalised Büchi Automata $\mathcal{GB}$ and Translation to (normal) Büchi Automata $\mathcal{B}$

Generalised Büchi Automata

A **generalised** Büchi automaton is defined as:

$$\mathcal{GB} = (Q, \delta, Q_0, \mathcal{F})$$

Q, δ, Q_0 as for standard Büchi automata

$\mathcal{F} = \{F_1, \dots, F_k\}$ is a **set of sets of accepting locations**
($F_i = \{f_{i1}, \dots, f_{im_i}\} \subseteq Q$)

Generalised Büchi Automata

A **generalised** Büchi automaton is defined as:

$$\mathcal{GB} = (Q, \delta, Q_0, \mathcal{F})$$

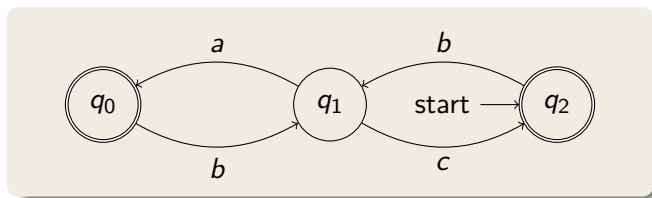
Q, δ, Q_0 as for standard Büchi automata

$\mathcal{F} = \{F_1, \dots, F_k\}$ is a **set of sets of accepting locations**
($F_i = \{f_{i1}, \dots, f_{im_i}\} \subseteq Q$)

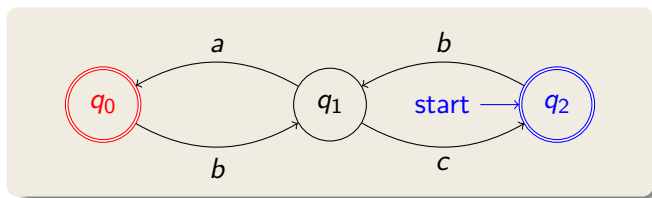
Definition (Acceptance for generalised Büchi automata)

A generalised Büchi automaton **accepts** an ω -word $w \in \Sigma^\omega$ iff
for every $i \in \{1, \dots, k\}$ **at least one** $q \in F_i$ is visited infinitely often.

Generalised vs. Normal Büchi Automata: Example

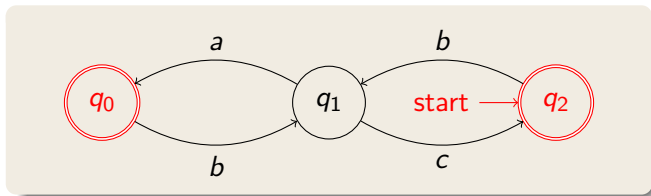


Generalised vs. Normal Büchi Automata: Example



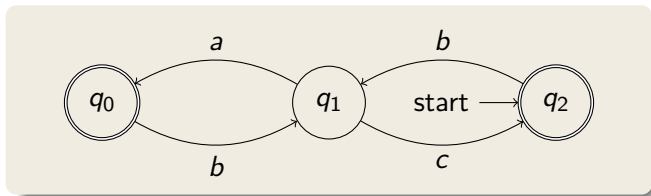
$\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$

Generalised vs. Normal Büchi Automata: Example



$\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$ different from normal $\mathcal{B}$ with $F = \{q_0, q_2\}$

Generalised vs. Normal Büchi Automata: Example

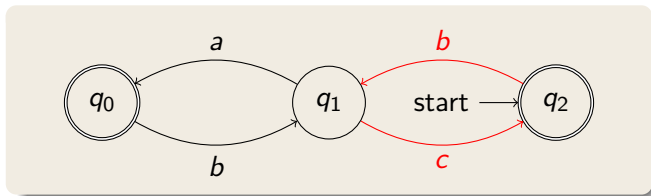


$\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$ different from normal $\mathcal{B}$ with $F = \{q_0, q_2\}$

Are the following ω -words accepted?

ω -word	$\mathcal{B}$	$\mathcal{GB}$
----------------	---------------	----------------

Generalised vs. Normal Büchi Automata: Example

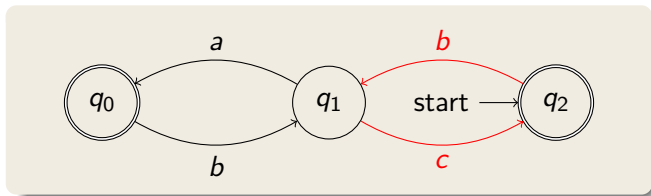


$\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$ different from normal $\mathcal{B}$ with $F = \{q_0, q_2\}$

Are the following ω -words accepted?

ω -word	$\mathcal{B}$	$\mathcal{GB}$
$(bc)^\omega$		
$(babc)^\omega$		

Generalised vs. Normal Büchi Automata: Example



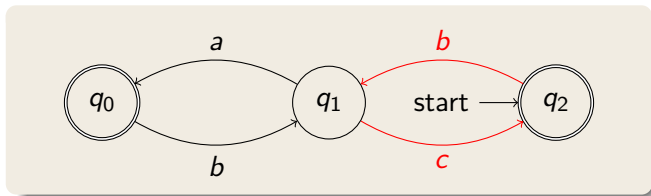
$\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$ different from normal $\mathcal{B}$ with $F = \{q_0, q_2\}$

Are the following ω -words accepted?

ω -word	$\mathcal{B}$	$\mathcal{GB}$
$(bc)^\omega$		
$(babc)^\omega$		

Poll

Generalised vs. Normal Büchi Automata: Example



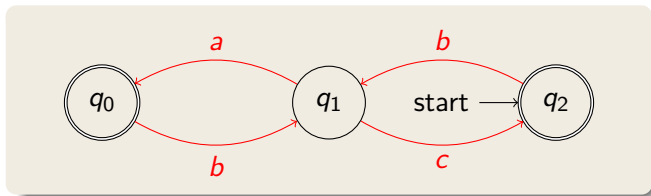
$\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$ different from normal $\mathcal{B}$ with $F = \{q_0, q_2\}$

Are the following ω -words accepted?

ω -word	$\mathcal{B}$	$\mathcal{GB}$
$(bc)^\omega$	✓	
$(babc)^\omega$		

Poll

Generalised vs. Normal Büchi Automata: Example



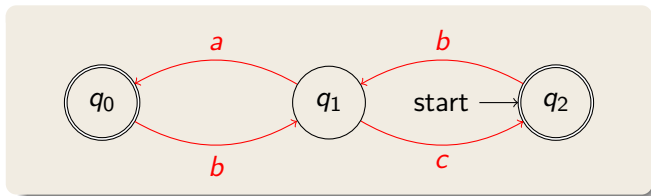
$\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$ different from normal $\mathcal{B}$ with $F = \{q_0, q_2\}$

Are the following ω -words accepted?

ω -word	$\mathcal{B}$	$\mathcal{GB}$
$(bc)^\omega$	✓	✗
$(babc)^\omega$		

Poll

Generalised vs. Normal Büchi Automata: Example



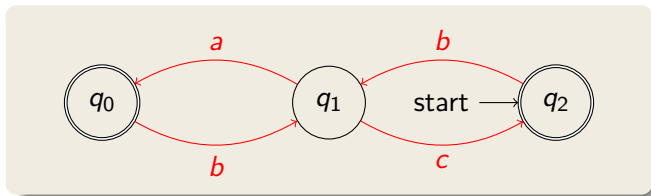
$\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$ different from normal $\mathcal{B}$ with $F = \{q_0, q_2\}$

Are the following ω -words accepted?

ω -word	$\mathcal{B}$	$\mathcal{GB}$
$(bc)^\omega$	✓	✗
$(babc)^\omega$	✓	

Poll

Generalised vs. Normal Büchi Automata: Example



$\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$ different from normal $\mathcal{B}$ with $F = \{q_0, q_2\}$

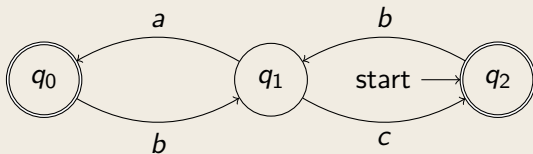
Are the following ω -words accepted?

ω -word	$\mathcal{B}$	$\mathcal{GB}$
$(bc)^\omega$	✓	✗
$(babc)^\omega$	✓	✓

Poll

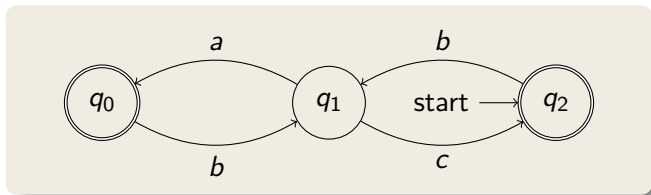
Translate Generalised to Normal Büchi Automata

$\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$:



Translate Generalised to Normal Büchi Automata

$\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$:

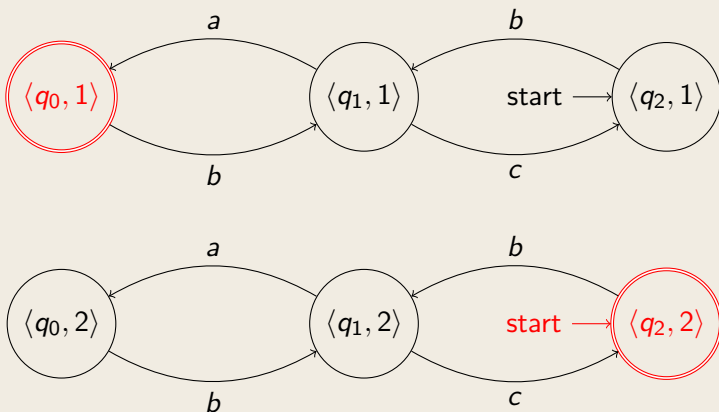


Construct $\mathcal{B}$ (different from last slide) which accepts the same words:

$$\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{GB})$$

Translate Generalised to Normal Büchi Automata

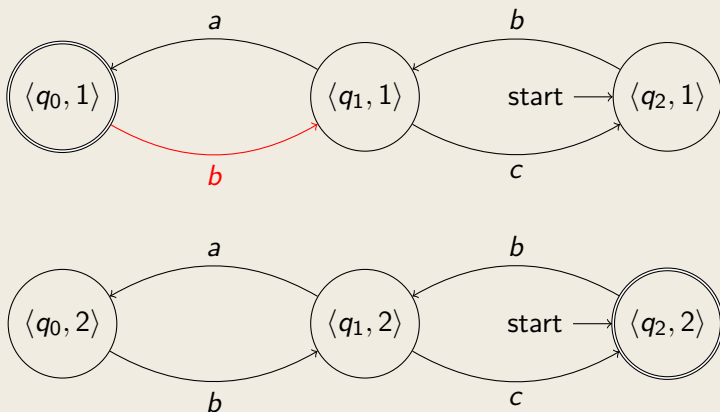
Construct $\mathcal{B}$ for $\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$:



One clone for each $F_i \in \mathcal{F}$

Translate Generalised to Normal Büchi Automata

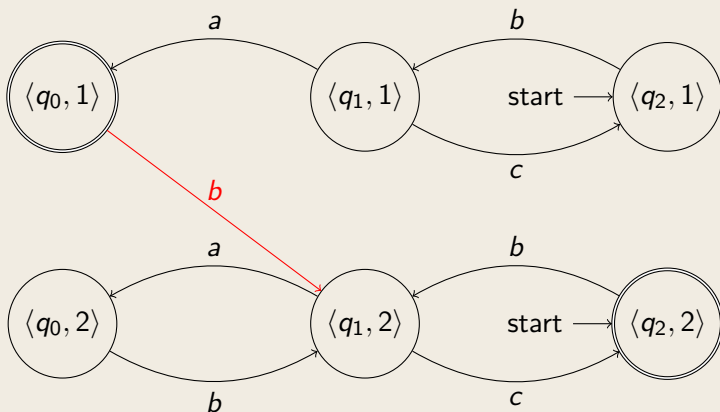
Construct $\mathcal{B}$ for $\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$:



Every **transition from " F_1 "** is redirected to "clone 2"

Translate Generalised to Normal Büchi Automata

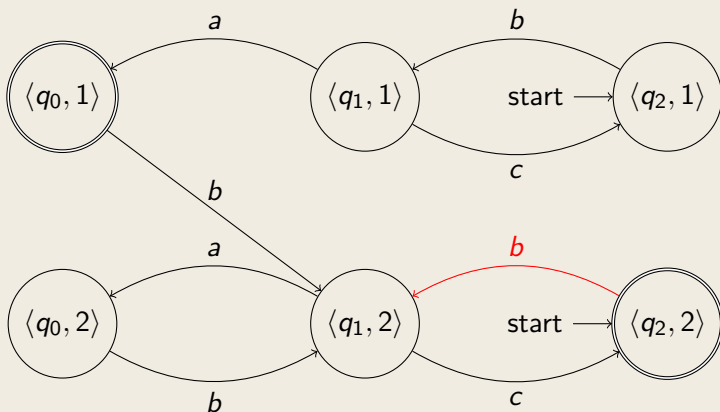
Construct $\mathcal{B}$ for $\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$:



Every transition from " F_1 " is **redirected to "clone 2"**

Translate Generalised to Normal Büchi Automata

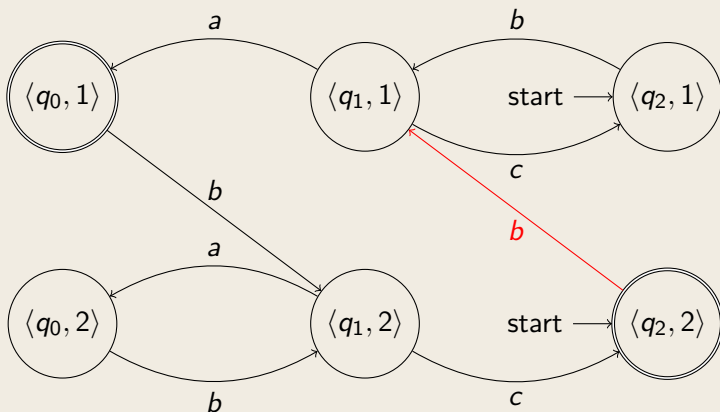
Construct $\mathcal{B}$ for $\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$:



Every **transition from " F_2 "** is redirected to "clone 1"

Translate Generalised to Normal Büchi Automata

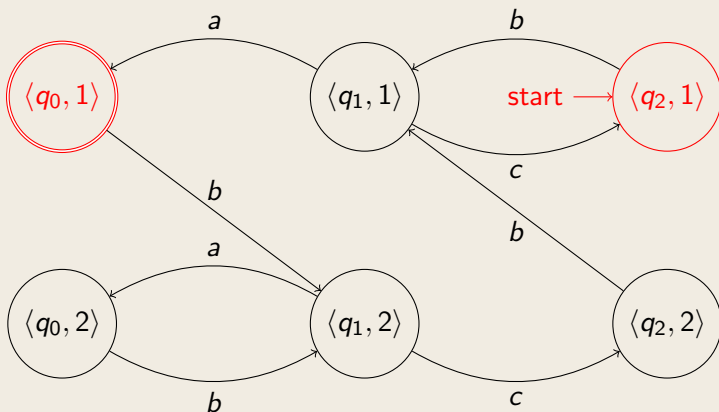
Construct $\mathcal{B}$ for $\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$:



Every transition from " F_2 " is **redirected to "clone 1"**

Translate Generalised to Normal Büchi Automata

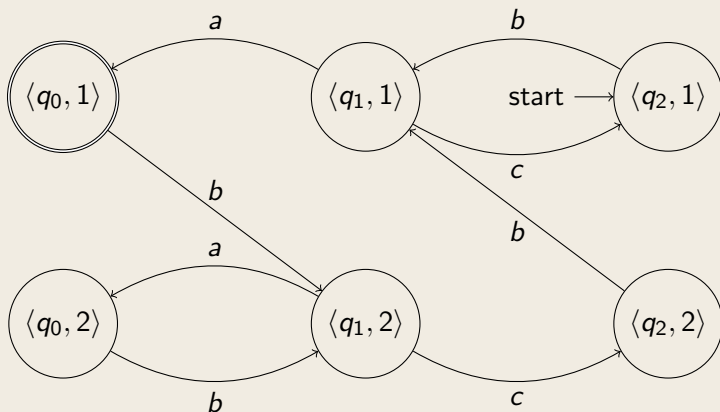
Construct $\mathcal{B}$ for $\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$:



Keep only **initial and accepting locations of “clone 1”**

Translate Generalised to Normal Büchi Automata

Construct $\mathcal{B}$ for $\mathcal{GB}$ with $\mathcal{F} = \{\overbrace{\{q_0\}}^{F_1}, \overbrace{\{q_2\}}^{F_2}\}$:



resulting *normal* Büchi automaton $\mathcal{B}$ with $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{GB})$

Translate Generalised to Normal Büchi Automata (formal)

Given **generalised** Büchi automaton

$$\mathcal{GB} = (Q, \delta, Q_0, \mathcal{F}) \quad \text{with } \mathcal{F} = \{F_1, \dots, F_k\}$$

Translate Generalised to Normal Büchi Automata (formal)

Given **generalised** Büchi automaton

$$\mathcal{GB} = (Q, \delta, Q_0, \mathcal{F}) \quad \text{with } \mathcal{F} = \{F_1, \dots, F_k\}$$

Equivalent **normal** Büchi automaton

$$\mathcal{B} = (Q', \delta', Q'_0, F') \quad \text{with}$$

Translate Generalised to Normal Büchi Automata (formal)

Given generalised Büchi automaton

$$\mathcal{GB} = (Q, \delta, Q_0, \mathcal{F}) \quad \text{with } \mathcal{F} = \{F_1, \dots, F_k\}$$

Equivalent normal Büchi automaton

$$\mathcal{B} = (Q', \delta', Q'_0, F') \quad \text{with}$$

- ▶ $Q' = Q \times \{1, \dots, k\}$

Translate Generalised to Normal Büchi Automata (formal)

Given generalised Büchi automaton

$$\mathcal{GB} = (Q, \delta, Q_0, \mathcal{F}) \quad \text{with } \mathcal{F} = \{F_1, \dots, F_k\}$$

Equivalent normal Büchi automaton

$$\mathcal{B} = (Q', \delta', Q'_0, F') \quad \text{with}$$

$$\blacktriangleright Q' = Q \times \{1, \dots, k\}$$

$$\blacktriangleright \delta'(\langle q, i \rangle, \sigma) = \begin{cases} \{\langle q', i \rangle \mid q' \in \delta(q, \sigma)\} & \text{if } q \notin F_i \\ \{\langle q', (i \bmod k) + 1 \rangle \mid q' \in \delta(q, \sigma)\} & \text{if } q \in F_i \end{cases}$$

Translate Generalised to Normal Büchi Automata (formal)

Given generalised Büchi automaton

$$\mathcal{GB} = (Q, \delta, Q_0, \mathcal{F}) \quad \text{with } \mathcal{F} = \{F_1, \dots, F_k\}$$

Equivalent normal Büchi automaton

$$\mathcal{B} = (Q', \delta', Q'_0, F') \quad \text{with}$$

- ▶ $Q' = Q \times \{1, \dots, k\}$
- ▶ $\delta'(\langle q, i \rangle, \sigma) = \begin{cases} \{\langle q', i \rangle \mid q' \in \delta(q, \sigma)\} & \text{if } q \notin F_i \\ \{\langle q', (i \bmod k) + 1 \rangle \mid q' \in \delta(q, \sigma)\} & \text{if } q \in F_i \end{cases}$
- ▶ $Q'_0 = \{\langle q, 1 \rangle \mid q \in Q_0\}$

Translate Generalised to Normal Büchi Automata (formal)

Given generalised Büchi automaton

$$\mathcal{GB} = (Q, \delta, Q_0, \mathcal{F}) \quad \text{with } \mathcal{F} = \{F_1, \dots, F_k\}$$

Equivalent normal Büchi automaton

$$\mathcal{B} = (Q', \delta', Q'_0, F') \quad \text{with}$$

- ▶ $Q' = Q \times \{1, \dots, k\}$
- ▶ $\delta'(\langle q, i \rangle, \sigma) = \begin{cases} \{\langle q', i \rangle \mid q' \in \delta(q, \sigma)\} & \text{if } q \notin F_i \\ \{\langle q', (i \bmod k) + 1 \rangle \mid q' \in \delta(q, \sigma)\} & \text{if } q \in F_i \end{cases}$
- ▶ $Q'_0 = \{\langle q, 1 \rangle \mid q \in Q_0\}$
- ▶ $F' = \{\langle q, 1 \rangle \mid q \in F_1\}$

Translate Generalised to Normal Büchi Automata (formal)

Given generalised Büchi automaton

$$\mathcal{GB} = (Q, \delta, Q_0, \mathcal{F}) \quad \text{with } \mathcal{F} = \{F_1, \dots, F_k\}$$

Equivalent normal Büchi automaton

$$\mathcal{B} = (Q', \delta', Q'_0, F') \quad \text{with}$$

- ▶ $Q' = Q \times \{1, \dots, k\}$
- ▶ $\delta'(\langle q, i \rangle, \sigma) = \begin{cases} \{\langle q', i \rangle \mid q' \in \delta(q, \sigma)\} & \text{if } q \notin F_i \\ \{\langle q', (i \bmod k) + 1 \rangle \mid q' \in \delta(q, \sigma)\} & \text{if } q \in F_i \end{cases}$
- ▶ $Q'_0 = \{\langle q, 1 \rangle \mid q \in Q_0\}$
- ▶ $F' = \{\langle q, 1 \rangle \mid q \in F_1\}$

Construction of a
Generalised Büchi Automaton
 $\mathcal{GB}_\phi$
for an
LTL-Formula ϕ

Focus on $\Box$ -free and $\Diamond$ -free LTL

- ▶ Following construction assumes formulas without $\Box$ and $\Diamond$.
- ▶ Only temporal modality is $\mathcal{U}$.
- ▶ $\Box$ can be removed using

$$\Box\phi \equiv \neg\Diamond\neg\phi$$

- ▶ $\Diamond$ can be removed using

$$\Diamond\phi \equiv \text{true } \mathcal{U}\phi$$

Theory and Example at Once

We introduce the general construction together with example.

Task:

construct

$$\mathcal{GB}_\phi$$

for

$$\phi \equiv rUs$$

Fischer-Ladner Closure

Fischer-Ladner closure of an LTL-formula ϕ

$$FL(\phi) = \{\varphi \mid \varphi \text{ is subformula or negated subformula of } \phi\}$$

Fischer-Ladner Closure

Fischer-Ladner closure of an LTL-formula ϕ

$$FL(\phi) = \{\varphi \mid \varphi \text{ is subformula or negated subformula of } \phi\}$$

($\neg\neg\varphi$ is identified with φ)

Fischer-Ladner Closure

Fischer-Ladner closure of an LTL-formula ϕ

$$FL(\phi) = \{\varphi \mid \varphi \text{ is subformula or negated subformula of } \phi\}$$

($\neg\neg\varphi$ is identified with φ)

Example

We want to translate $\phi \equiv r\mathcal{U}s$

Fischer-Ladner Closure

Fischer-Ladner closure of an LTL-formula ϕ

$$FL(\phi) = \{\varphi \mid \varphi \text{ is subformula or negated subformula of } \phi\}$$

($\neg\neg\varphi$ is identified with φ)

Example

We want to translate $\phi \equiv r\mathcal{U}s$

$$FL(r\mathcal{U}s) = \{r, \neg r, s, \neg s, r\mathcal{U}s, \neg(r\mathcal{U}s)\}$$

$\mathcal{GB}_\phi$ -Construction: Locations

Locations of $\mathcal{GB}_\phi$ are $Q \subseteq 2^{FL(\phi)}$ where each $q \in Q$ satisfies:

$\mathcal{GB}_\phi$ -Construction: Locations

Locations of $\mathcal{GB}_\phi$ are $Q \subseteq 2^{FL(\phi)}$ where each $q \in Q$ satisfies:

consistent, total ▶ $\psi \in FL(\phi)$: *exactly* one of ψ and $\neg\psi$ in q

$\mathcal{GB}_\phi$ -Construction: Locations

Locations of $\mathcal{GB}_\phi$ are $Q \subseteq 2^{FL(\phi)}$ where each $q \in Q$ satisfies:

- consistent, total** ▶ $\psi \in FL(\phi)$: *exactly* one of ψ and $\neg\psi$ in q
- downward closed** ▶ $\psi_1 \wedge \psi_2 \in q$: $\psi_1 \in q$ and $\psi_2 \in q$

$\mathcal{GB}_\phi$ -Construction: Locations

Locations of $\mathcal{GB}_\phi$ are $Q \subseteq 2^{FL(\phi)}$ where each $q \in Q$ satisfies:

- consistent, total** ▶ $\psi \in FL(\phi)$: *exactly* one of ψ and $\neg\psi$ in q
- downward closed** ▶ $\psi_1 \wedge \psi_2 \in q$: $\psi_1 \in q$ and $\psi_2 \in q$
- ▶ $\psi_1 \vee \psi_2 \in q$: $\psi_1 \in q$ or $\psi_2 \in q$

$\mathcal{GB}_\phi$ -Construction: Locations

Locations of $\mathcal{GB}_\phi$ are $Q \subseteq 2^{FL(\phi)}$ where each $q \in Q$ satisfies:

- consistent, total** ▶ $\psi \in FL(\phi)$: *exactly* one of ψ and $\neg\psi$ in q
- downward closed** ▶ $\psi_1 \wedge \psi_2 \in q$: $\psi_1 \in q$ and $\psi_2 \in q$
- ▶ $\psi_1 \vee \psi_2 \in q$: $\psi_1 \in q$ or $\psi_2 \in q$
- ▶ $\psi_1 \rightarrow \psi_2 \in q$: $\neg\psi_1 \in q$ or $\psi_2 \in q$

$\mathcal{GB}_\phi$ -Construction: Locations

Locations of $\mathcal{GB}_\phi$ are $Q \subseteq 2^{FL(\phi)}$ where each $q \in Q$ satisfies:

- consistent, total** ▶ $\psi \in FL(\phi)$: *exactly* one of ψ and $\neg\psi$ in q
- downward closed** ▶ $\psi_1 \wedge \psi_2 \in q$: $\psi_1 \in q$ and $\psi_2 \in q$
 ▶ $\psi_1 \vee \psi_2 \in q$: $\psi_1 \in q$ or $\psi_2 \in q$
 ▶ $\psi_1 \rightarrow \psi_2 \in q$: $\neg\psi_1 \in q$ or $\psi_2 \in q$
- until-consistent** ▶ $\psi_1 \mathcal{U} \psi_2 \in q$ then $\psi_1 \in q$ or $\psi_2 \in q$

$\mathcal{GB}_\phi$ -Construction: Locations

Locations of $\mathcal{GB}_\phi$ are $Q \subseteq 2^{FL(\phi)}$ where each $q \in Q$ satisfies:

- consistent, total** ▶ $\psi \in FL(\phi)$: *exactly* one of ψ and $\neg\psi$ in q
- downward closed** ▶ $\psi_1 \wedge \psi_2 \in q$: $\psi_1 \in q$ and $\psi_2 \in q$
▶ $\psi_1 \vee \psi_2 \in q$: $\psi_1 \in q$ or $\psi_2 \in q$
▶ $\psi_1 \rightarrow \psi_2 \in q$: $\neg\psi_1 \in q$ or $\psi_2 \in q$
- until-consistent** ▶ $\psi_1 \mathcal{U} \psi_2 \in q$ then $\psi_1 \in q$ or $\psi_2 \in q$
▶ $\neg(\psi_1 \mathcal{U} \psi_2) \in q$ then $\neg\psi_2 \in q$

$\mathcal{B}_\phi$ -Construction: Locations

consistent, total	$\in Q$
$\{\neg(r\mathcal{U}s), \neg r, \neg s\}$	
$\{\neg(r\mathcal{U}s), \neg r, s\}$	
$\{\neg(r\mathcal{U}s), r, \neg s\}$	
$\{\neg(r\mathcal{U}s), r, s\}$	
$\{r\mathcal{U}s, \neg r, \neg s\}$	
$\{r\mathcal{U}s, \neg r, s\}$	
$\{r\mathcal{U}s, r, \neg s\}$	
$\{r\mathcal{U}s, r, s\}$	

$\mathcal{B}_\phi$ -Construction: Locations

consistent, total	$\in Q$
$\{\neg(r\mathcal{U}s), \neg r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), \neg r, s\}$	
$\{\neg(r\mathcal{U}s), r, \neg s\}$	
$\{\neg(r\mathcal{U}s), r, s\}$	
$\{r\mathcal{U}s, \neg r, \neg s\}$	
$\{r\mathcal{U}s, \neg r, s\}$	
$\{r\mathcal{U}s, r, \neg s\}$	
$\{r\mathcal{U}s, r, s\}$	

$\mathcal{B}_\phi$ -Construction: Locations

consistent, total	$\in Q$
$\{\neg(r\mathcal{U}s), \neg r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), \neg r, s\}$	✗
$\{\neg(r\mathcal{U}s), r, \neg s\}$	
$\{\neg(r\mathcal{U}s), r, s\}$	
$\{r\mathcal{U}s, \neg r, \neg s\}$	
$\{r\mathcal{U}s, \neg r, s\}$	
$\{r\mathcal{U}s, r, \neg s\}$	
$\{r\mathcal{U}s, r, s\}$	

$\mathcal{B}_\phi$ -Construction: Locations

consistent, total	$\in Q$
$\{\neg(r\mathcal{U}s), \neg r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), \neg r, s\}$	✗
$\{\neg(r\mathcal{U}s), r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), r, s\}$	
$\{r\mathcal{U}s, \neg r, \neg s\}$	
$\{r\mathcal{U}s, \neg r, s\}$	
$\{r\mathcal{U}s, r, \neg s\}$	
$\{r\mathcal{U}s, r, s\}$	

$\mathcal{B}_\phi$ -Construction: Locations

consistent, total	$\in Q$
$\{\neg(r\mathcal{U}s), \neg r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), \neg r, s\}$	✗
$\{\neg(r\mathcal{U}s), r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), r, s\}$	✗
$\{r\mathcal{U}s, \neg r, \neg s\}$	
$\{r\mathcal{U}s, \neg r, s\}$	
$\{r\mathcal{U}s, r, \neg s\}$	
$\{r\mathcal{U}s, r, s\}$	

$\mathcal{B}_\phi$ -Construction: Locations

consistent, total	$\in Q$
$\{\neg(r\mathcal{U}s), \neg r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), \neg r, s\}$	✗
$\{\neg(r\mathcal{U}s), r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), r, s\}$	✗
$\{r\mathcal{U}s, \neg r, \neg s\}$	✗
$\{r\mathcal{U}s, \neg r, s\}$	
$\{r\mathcal{U}s, r, \neg s\}$	
$\{r\mathcal{U}s, r, s\}$	

$\mathcal{B}_\phi$ -Construction: Locations

consistent, total	$\in Q$
$\{\neg(r\mathcal{U}s), \neg r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), \neg r, s\}$	✗
$\{\neg(r\mathcal{U}s), r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), r, s\}$	✗
$\{r\mathcal{U}s, \neg r, \neg s\}$	✗
$\{r\mathcal{U}s, \neg r, s\}$	✓
$\{r\mathcal{U}s, r, \neg s\}$	
$\{r\mathcal{U}s, r, s\}$	

$\mathcal{B}_\phi$ -Construction: Locations

consistent, total	$\in Q$
$\{\neg(r\mathcal{U}s), \neg r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), \neg r, s\}$	✗
$\{\neg(r\mathcal{U}s), r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), r, s\}$	✗
$\{r\mathcal{U}s, \neg r, \neg s\}$	✗
$\{r\mathcal{U}s, \neg r, s\}$	✓
$\{r\mathcal{U}s, r, \neg s\}$	✓
$\{r\mathcal{U}s, r, s\}$	

$\mathcal{B}_\phi$ -Construction: Locations

consistent, total	$\in Q$
$\{\neg(r\mathcal{U}s), \neg r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), \neg r, s\}$	✗
$\{\neg(r\mathcal{U}s), r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), r, s\}$	✗
$\{r\mathcal{U}s, \neg r, \neg s\}$	✗
$\{r\mathcal{U}s, \neg r, s\}$	✓
$\{r\mathcal{U}s, r, \neg s\}$	✓
$\{r\mathcal{U}s, r, s\}$	✓

$\mathcal{B}_\phi$ -Construction: Locations

consistent, total	$\in Q$
$\{\neg(r\mathcal{U}s), \neg r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), \neg r, s\}$	✗
$\{\neg(r\mathcal{U}s), r, \neg s\}$	✓
$\{\neg(r\mathcal{U}s), r, s\}$	✗
$\{r\mathcal{U}s, \neg r, \neg s\}$	✗
$\{r\mathcal{U}s, \neg r, s\}$	✓
$\{r\mathcal{U}s, r, \neg s\}$	✓
$\{r\mathcal{U}s, r, s\}$	✓

Locations of $\mathcal{B}_\phi$ are sets of formulas which can be *simultaneously* true

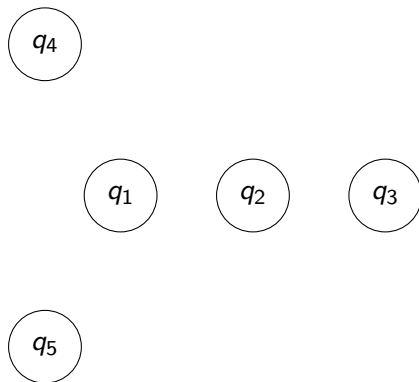
$\mathcal{B}_\phi$ -Construction: Transitions

$$\underbrace{\{rUs, \neg r, s\}}_{q_1}, \underbrace{\{rUs, r, \neg s\}}_{q_2}, \underbrace{\{rUs, r, s\}}_{q_3}, \underbrace{\{\neg(rUs), r, \neg s\}}_{q_4}, \underbrace{\{\neg(rUs), \neg r, \neg s\}}_{q_5}$$



$\mathcal{B}_\phi$ -Construction: Transitions

$$\underbrace{\{rUs, \neg r, s\}}_{q_1}, \underbrace{\{rUs, r, \neg s\}}_{q_2}, \underbrace{\{rUs, r, s\}}_{q_3}, \underbrace{\{\neg(rUs), r, \neg s\}}_{q_4}, \underbrace{\{\neg(rUs), \neg r, \neg s\}}_{q_5}$$



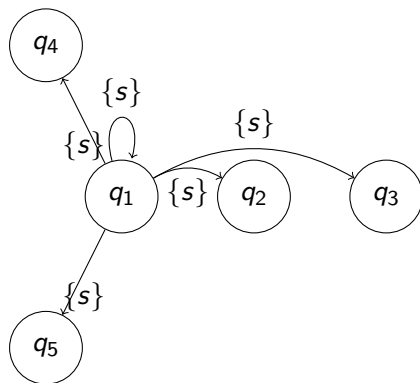
Transitions $(q, \alpha, q') \in \delta_\phi$:

such that

1. $\alpha = q \cap AP$
(AP : atomic propositions)
2. If $\psi_1 \mathcal{U} \psi_2 \in q$ and $\neg \psi_2 \in q$ then $\psi_1 \mathcal{U} \psi_2 \in q'$
3. If $\neg(\psi_1 \mathcal{U} \psi_2) \in q$ and $\psi_1 \in q$ then $\neg(\psi_1 \mathcal{U} \psi_2) \in q'$

$\mathcal{B}_\phi$ -Construction: Transitions

$$\underbrace{\{rUs, \neg r, s\}}_{q_1}, \underbrace{\{rUs, r, \neg s\}}_{q_2}, \underbrace{\{rUs, r, s\}}_{q_3}, \underbrace{\{\neg(rUs), r, \neg s\}}_{q_4}, \underbrace{\{\neg(rUs), \neg r, \neg s\}}_{q_5}$$



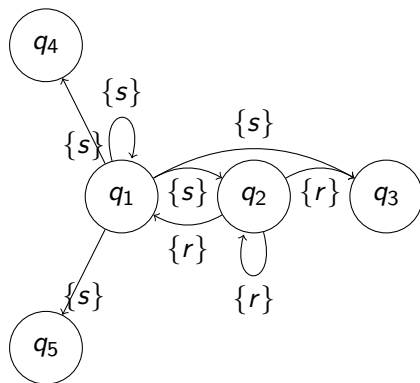
Transitions $(q, \alpha, q') \in \delta_\phi$:

such that

1. $\alpha = q \cap AP$
(AP : atomic propositions)
2. If $\psi_1 \mathcal{U} \psi_2 \in q$ and $\neg \psi_2 \in q$ then $\psi_1 \mathcal{U} \psi_2 \in q'$
3. If $\neg(\psi_1 \mathcal{U} \psi_2) \in q$ and $\psi_1 \in q$ then $\neg(\psi_1 \mathcal{U} \psi_2) \in q'$

$\mathcal{B}_\phi$ -Construction: Transitions

$$\underbrace{\{rUs, \neg r, s\}}_{q_1}, \underbrace{\{rUs, r, \neg s\}}_{q_2}, \underbrace{\{rUs, r, s\}}_{q_3}, \underbrace{\{\neg(rUs), r, \neg s\}}_{q_4}, \underbrace{\{\neg(rUs), \neg r, \neg s\}}_{q_5}$$



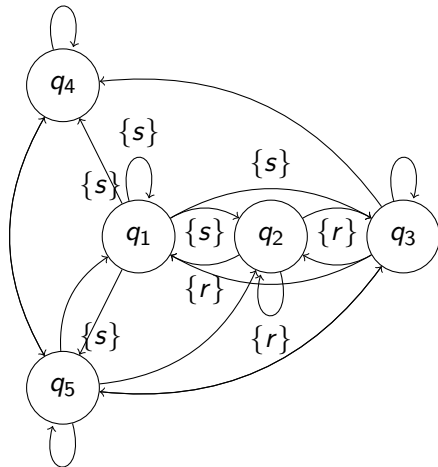
Transitions $(q, \alpha, q') \in \delta_\phi$:

such that

1. $\alpha = q \cap AP$
(AP : atomic propositions)
2. If $\psi_1 \mathcal{U} \psi_2 \in q$ and $\neg \psi_2 \in q$ then $\psi_1 \mathcal{U} \psi_2 \in q'$
3. If $\neg(\psi_1 \mathcal{U} \psi_2) \in q$ and $\psi_1 \in q$ then $\neg(\psi_1 \mathcal{U} \psi_2) \in q'$

$\mathcal{B}_\phi$ -Construction: Transitions

$$\underbrace{\{rUs, \neg r, s\}}_{q_1}, \underbrace{\{rUs, r, \neg s\}}_{q_2}, \underbrace{\{rUs, r, s\}}_{q_3}, \underbrace{\{\neg(rUs), r, \neg s\}}_{q_4}, \underbrace{\{\neg(rUs), \neg r, \neg s\}}_{q_5}$$



Transitions $(q, \alpha, q') \in \delta_\phi$:

such that

1. $\alpha = q \cap AP$
(AP : atomic propositions)
2. If $\psi_1 \mathcal{U} \psi_2 \in q$ and $\neg \psi_2 \in q$ then $\psi_1 \mathcal{U} \psi_2 \in q'$
3. If $\neg(\psi_1 \mathcal{U} \psi_2) \in q$ and $\psi_1 \in q$ then $\neg(\psi_1 \mathcal{U} \psi_2) \in q'$

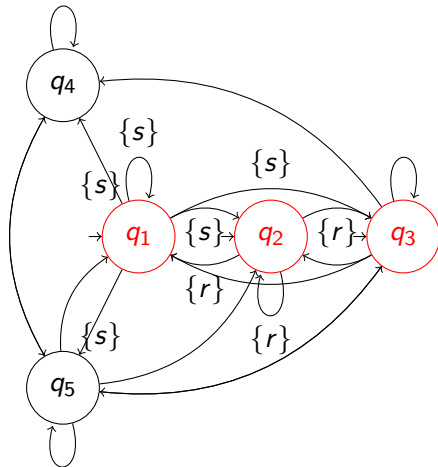
(skipping some labels to save space)

$\mathcal{B}_\phi$ -Construction: Transitions

$$\underbrace{\{rUs, \neg r, s\}}_{q_1}, \underbrace{\{rUs, r, \neg s\}}_{q_2}, \underbrace{\{rUs, r, s\}}_{q_3}, \underbrace{\{\neg(rUs), r, \neg s\}}_{q_4}, \underbrace{\{\neg(rUs), \neg r, \neg s\}}_{q_5}$$

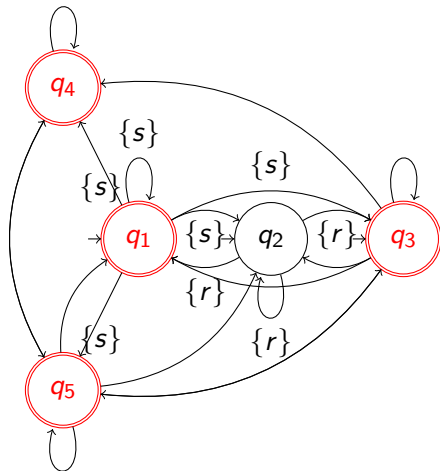
Initial locations

$$q \in I_\phi \text{ iff } \phi \in q$$



$\mathcal{B}_\phi$ -Construction: Transitions

$$\underbrace{\{rUs, \neg r, s\}}_{q_1}, \underbrace{\{rUs, r, \neg s\}}_{q_2}, \underbrace{\{rUs, r, s\}}_{q_3}, \underbrace{\{\neg(rUs), r, \neg s\}}_{q_4}, \underbrace{\{\neg(rUs), \neg r, \neg s\}}_{q_5}$$



Initial locations

$$q \in I_\phi \text{ iff } \phi \in q$$

Accepting locations

$$\mathcal{F} = \{F_1, \dots, F_n\}$$

- One F_i for each $\psi_{i1} \mathcal{U} \psi_{i2} \in FL(\phi)$.
here: $\mathcal{F} = \{F_1\}$
- F_i set of locations that do *not* contain $\psi_{i1} \mathcal{U} \psi_{i2}$ **or** that contain ψ_{i2} .
here: $F_1 = \{q_1, q_3, q_4, q_5\}$

Remarks on Generalized Büchi Automata

- ▶ Construction gives exponential number of states in $|\phi|$
- ▶ Satisfiability checking of LTL is PSPACE-complete
- ▶ There exist (more complex) constructions that reduce number of required states
 - ▶ One of these is used in SPIN, which moreover computes the states lazily

Practical Temporal MC Revisited

```
byte sem = 2;

active [4] proctype P() {
  do
    :: printf("Non_critical_section_P\n");
    atomic {
      sem > 0;
      sem--
    }
    printf("Critical_section_P\n");
    sem++
  od
}
```

Practical Temporal MC Revisited

```
byte sem = 2;

active [4] proctype P() {
  do
    :: printf("Non_critical_section_P\n");
    atomic {
      sem > 0;
      sem--
    }
    printf("Critical_section_P\n");
    sem++
  od
}
```

► $\Box(\text{sem} \leq 2)$? Poll

Practical Temporal MC Revisited

```
byte sem = 2;

active [4] proctype P() {
  do
    :: printf("Non_critical_section_P\n");
    atomic {
      sem > 0;
      sem--
    }
    printf("Critical_section_P\n");
    sem++
  od
}
```

- ▶ $\Box(\text{sem} \leq 2) ?$ Poll
- ▶ $\Box\Diamond(\text{sem} = 2) ?$ Poll

Practical Temporal MC Revisited

```
byte sem = 2;

active [4] proctype P() {
  do
    :: printf("Non_critical_section_P\n");
    atomic {
      sem > 0;
      sem--
    }
    printf("Critical_section_P\n");
    sem++
  od
}
```

- ▶ $\Box(\text{sem} \leq 2) ?$ Poll
- ▶ $\Box\Diamond(\text{sem} = 2) ?$ Poll
- ▶ $\Box\Diamond(\text{sem} = 0) ?$

Practical Temporal MC Revisited

```
byte sem = 2;

active [4] proctype P() {
  do
    :: printf("Non_critical_section_P\n");
      atomic {
        sem > 0;
        sem--
      }
    printf("Critical_section_P\n");
    sem++
  od
}
```

- ▶ $\Box(\text{sem} \leq 2) ?$ Poll
- ▶ $\Box\Diamond(\text{sem} = 2) ?$ Poll
- ▶ $\Box\Diamond(\text{sem} = 0) ?$
- ▶ $\Box\Diamond(\text{sem} = 2 \vee \text{sem} = 0) ?$ Poll

Practical Temporal MC Revisited

```
byte sem = 2;

active [4] proctype P() {
  do
    :: printf("Non_critical_section_P\n");
      atomic {
        sem > 0;
        sem--
      }
    printf("Critical_section_P\n");
    sem++
  od
}
```

- ▶ $\Box(\text{sem} \leq 2) ?$ Poll
- ▶ $\Box\Diamond(\text{sem} = 2) ?$ Poll
- ▶ $\Box\Diamond(\text{sem} = 0) ?$
- ▶ $\Box\Diamond(\text{sem} = 2 \vee \text{sem} = 0) ?$ Poll

Practical Temporal MC Revisited

```
byte sem = 2;

active [4] proctype P() {
  do
    :: printf("Non_critical_section_P\n");
      atomic {
        sem > 0;
        sem--;
      }
    printf("Critical_section_P\n");
    sem++;
  od
}
```

- ▶ $\Box(\text{sem} \leq 2)$? Poll
- ▶ $\Box\Diamond(\text{sem} = 2)$? Poll
- ▶ $\Box\Diamond(\text{sem} = 0)$?
- ▶ $\Box\Diamond(\text{sem} = 2 \vee \text{sem} = 0)$? Poll

Ask SPIN

Part II

Starting First-order Logic

Motivation for Introducing First-Order Logic

- 1) We specify JAVA programs with **Java Modeling Language (JML)**

JML combines

- ▶ JAVA expressions
- ▶ **First-Order Logic (FOL)**

- 2) We verify JAVA programs using **Dynamic Logic**

Dynamic Logic combines

- ▶ **First-Order Logic (FOL)**
- ▶ JAVA programs

FOL: Language and Calculus

We introduce:

- ▶ FOL as a language
- ▶ Sequent calculus for proving FOL formulas
- ▶ KeY system as propositional, and first-order, prover (for now)
- ▶ Formal semantics

First-Order Logic: Signature

Signature

A first-order signature Σ consists of

- ▶ a set T_Σ of types
- ▶ a set F_Σ of function symbols
- ▶ a set P_Σ of predicate symbols
- ▶ a typing α_Σ

First-Order Logic: Signature

Signature

A first-order signature Σ consists of

- ▶ a set T_Σ of types
- ▶ a set F_Σ of function symbols
- ▶ a set P_Σ of predicate symbols
- ▶ a typing α_Σ

Intuitively, the typing α_Σ determines

- ▶ for each function and predicate symbol:
 - ▶ its arity, i.e., number of arguments
 - ▶ its argument types
- ▶ for each function symbol its result type.

First-Order Logic: Signature

Signature

A first-order signature Σ consists of

- ▶ a set T_Σ of types
- ▶ a set F_Σ of function symbols
- ▶ a set P_Σ of predicate symbols
- ▶ a typing α_Σ

Intuitively, the typing α_Σ determines

- ▶ for each function and predicate symbol:
 - ▶ its arity, i.e., number of arguments
 - ▶ its argument types
- ▶ for each function symbol its result type.

Formally:

- ▶ $\alpha_\Sigma(p) \in T_\Sigma^*$ for all $p \in P_\Sigma$ (arity of p is $|\alpha_\Sigma(p)|$)
- ▶ $\alpha_\Sigma(f) \in T_\Sigma^* \times T_\Sigma$ for all $f \in F_\Sigma$ (arity of f is $|\alpha_\Sigma(f)| - 1$)

Example Signature Σ_1 + Constants

$$T_{\Sigma_1} = \{\text{int}\},$$

$$F_{\Sigma_1} = \{+, -\} \cup \{\dots, -2, -1, 0, 1, 2, \dots\},$$

$$P_{\Sigma_1} = \{<\}$$

$$\alpha_{\Sigma_1}(<) = (\text{int}, \text{int})$$

$$\alpha_{\Sigma_1}(+) = \alpha_{\Sigma_1}(-) = (\text{int}, \text{int}, \text{int})$$

$$\alpha_{\Sigma_1}(0) = \alpha_{\Sigma_1}(1) = \alpha_{\Sigma_1}(-1) = \dots = (\text{int})$$

Constant Symbols

A function symbol f with $|\alpha_{\Sigma_1}(f)| = 1$ (i.e., with arity 0) is called *constant symbol*.

Here, the constant symbols are: $\dots, -2, -1, 0, 1, 2, \dots$

Syntax of First-Order Logic: Signature Cont'd

Type declaration of signature symbols

- ▶ Write τx ; to declare variable x of type τ
- ▶ Write $p(\tau_1, \dots, \tau_r)$; for $\alpha(p) = (\tau_1, \dots, \tau_r)$
- ▶ Write $\tau f(\tau_1, \dots, \tau_r)$; for $\alpha(f) = (\tau_1, \dots, \tau_r, \tau)$

$r = 0$ is allowed, then write f and p instead of $f()$ and $p()$.

Example

Variables `integerArray a; int i;`

Predicate Symbols `isEmpty(List); alertOn;`

Function Symbols `int arrayLookup(int); Object o;`

Example Signature Σ_1 + Notation

Typing of Signature:

$$\alpha_{\Sigma_1}(<) = (\text{int}, \text{int})$$

$$\alpha_{\Sigma_1}(+) = \alpha_{\Sigma_1}(-) = (\text{int}, \text{int}, \text{int})$$

$$\alpha_{\Sigma_1}(0) = \alpha_{\Sigma_1}(1) = \alpha_{\Sigma_1}(-1) = \dots = (\text{int})$$

can alternatively be written as:

`<(int,int);`

`int +(int,int);`

`int 0; int 1; int -1; ...`

First-Order Terms

We assume a set V of variables ($V \cap (F_\Sigma \cup P_\Sigma) = \emptyset$).
Each $v \in V$ has a unique type $\alpha_\Sigma(v) \in T_\Sigma$.

First-Order Terms

We assume a set V of variables ($V \cap (F_\Sigma \cup P_\Sigma) = \emptyset$).

Each $v \in V$ has a unique type $\alpha_\Sigma(v) \in T_\Sigma$.

Terms are defined recursively:

Terms

A first-order term of type $\tau \in T_\Sigma$

- ▶ is either a variable of type τ , or
- ▶ has the form $f(t_1, \dots, t_n)$,
where $f \in F_\Sigma$ has result type τ , and each t_i is term of the correct type, following the typing α_Σ of f .

If f is a constant symbol, the term is written f , instead of $f()$.

Terms over Signature Σ_1

Example terms over Σ_1 :

(assume variables `int  $v_1$ ; int  $v_2$ ;`)

Terms over Signature Σ_1

Example terms over Σ_1 :

(assume variables $\text{int } v_1; \text{ int } v_2;$)

- ▶ -7
- ▶ $+(-2, 99)$
- ▶ $-(7, 8)$
- ▶ $+(-(7, 8), 1)$
- ▶ $+(-(v_1, 8), v_2)$

Terms over Signature Σ_1

Example terms over Σ_1 :

(assume variables $\text{int } v_1; \text{ int } v_2;$)

- ▶ -7
- ▶ $+(-2, 99)$
- ▶ $-(7, 8)$
- ▶ $+(-(7, 8), 1)$
- ▶ $+(-(v_1, 8), v_2)$

Our variant of FOL allows infix notation for common functions:

- ▶ $-2 + 99$
- ▶ $7 - 8$
- ▶ $(7 - 8) + 1$
- ▶ $(v_1 - 8) + v_2$

Atomic Formulas

Given a signature Σ .

An atomic formula has either of the forms

- ▶ *true*
- ▶ *false*
- ▶ $t_1 = t_2$ (“equality”),
where t_1 and t_2 are first-order terms of the same type.
- ▶ $p(t_1, \dots, t_n)$ (“predicate”),
where $p \in P_\Sigma$, and each t_i is term of the correct type,
following the typing α_Σ of p .

Atomic Formulas over Signature Σ_1

Example formulas over Σ_1 :
(assume variable `int v`;))

- ▶ $7 = 8$
- ▶ $<(7, 8)$
- ▶ $<(-2 - v, 99)$
- ▶ $<(v, v + 1)$

Atomic Formulas over Signature Σ_1

Example formulas over Σ_1 :
(assume variable `int v`;

- ▶ $7 = 8$
- ▶ $<(7, 8)$
- ▶ $<(-2 - v, 99)$
- ▶ $<(v, v + 1)$

Our variant of FOL allows infix notation for common predicates:

- ▶ $7 < 8$
- ▶ $-2 - v < 99$
- ▶ $v < v + 1$

First-Order Formulas

Formulas

- ▶ each atomic formula is a formula
- ▶ with ϕ and ψ formulas, x a variable, and τ a type, the following are also formulas:
 - ▶ $\neg\phi$ (*“not ϕ ”*)
 - ▶ $\phi \wedge \psi$ (*“ ϕ and ψ ”*)
 - ▶ $\phi \vee \psi$ (*“ ϕ or ψ ”*)
 - ▶ $\phi \rightarrow \psi$ (*“ ϕ implies ψ ”*)
 - ▶ $\phi \leftrightarrow \psi$ (*“ ϕ is equivalent to ψ ”*)

First-Order Formulas

Formulas

- ▶ each atomic formula is a formula
- ▶ with ϕ and ψ formulas, x a variable, and τ a type, the following are also formulas:
 - ▶ $\neg\phi$ (*“not ϕ ”*)
 - ▶ $\phi \wedge \psi$ (*“ ϕ and ψ ”*)
 - ▶ $\phi \vee \psi$ (*“ ϕ or ψ ”*)
 - ▶ $\phi \rightarrow \psi$ (*“ ϕ implies ψ ”*)
 - ▶ $\phi \leftrightarrow \psi$ (*“ ϕ is equivalent to ψ ”*)
 - ▶ $\forall \tau \ x; \phi$ (*“for all x of type τ holds ϕ ”*)
 - ▶ $\exists \tau \ x; \phi$ (*“there exists an x of type τ such that ϕ ”*)

First-Order Formulas

Formulas

- ▶ each atomic formula is a formula
- ▶ with ϕ and ψ formulas, x a variable, and τ a type, the following are also formulas:
 - ▶ $\neg\phi$ (“not ϕ ”)
 - ▶ $\phi \wedge \psi$ (“ ϕ and ψ ”)
 - ▶ $\phi \vee \psi$ (“ ϕ or ψ ”)
 - ▶ $\phi \rightarrow \psi$ (“ ϕ implies ψ ”)
 - ▶ $\phi \leftrightarrow \psi$ (“ ϕ is equivalent to ψ ”)
 - ▶ $\forall \tau x; \phi$ (“for all x of type τ holds ϕ ”)
 - ▶ $\exists \tau x; \phi$ (“there exists an x of type τ such that ϕ ”)

In $\forall \tau x; \phi$ and $\exists \tau x; \phi$ the variable x is ‘bound’ (i.e., ‘not free’).

First-Order Formulas

Formulas

- ▶ each atomic formula is a formula
- ▶ with ϕ and ψ formulas, x a variable, and τ a type, the following are also formulas:
 - ▶ $\neg\phi$ (“not ϕ ”)
 - ▶ $\phi \wedge \psi$ (“ ϕ and ψ ”)
 - ▶ $\phi \vee \psi$ (“ ϕ or ψ ”)
 - ▶ $\phi \rightarrow \psi$ (“ ϕ implies ψ ”)
 - ▶ $\phi \leftrightarrow \psi$ (“ ϕ is equivalent to ψ ”)
 - ▶ $\forall \tau x; \phi$ (“for all x of type τ holds ϕ ”)
 - ▶ $\exists \tau x; \phi$ (“there exists an x of type τ such that ϕ ”)

In $\forall \tau x; \phi$ and $\exists \tau x; \phi$ the variable x is ‘bound’ (i.e., ‘not free’).
Formulas with no free variable are ‘closed’.

First-order Formulas: Examples

(signatures/types left out here)

Example (There are at least two elements)

First-order Formulas: Examples

(signatures/types left out here)

Example (There are at least two elements)

$$\exists x, y; \neg(x = y)$$

First-order Formulas: Examples

(signatures/types left out here)

Example (Strict partial order)



First-order Formulas: Examples

(signatures/types left out here)

Example (Strict partial order)

Irreflexivity $\forall x; \neg(x < x)$

Asymmetry $\forall x; \forall y; (x < y \rightarrow \neg(y < x))$

Transitivity $\forall x; \forall y; \forall z;$
 $(x < y \wedge y < z \rightarrow x < z)$

First-order Formulas: Examples

(signatures/types left out here)

Example (Strict partial order)

Irreflexivity $\forall x; \neg(x < x)$

Asymmetry $\forall x; \forall y; (x < y \rightarrow \neg(y < x))$

Transitivity $\forall x; \forall y; \forall z;$
 $(x < y \wedge y < z \rightarrow x < z)$

(Is any of the three formulas redundant?)

Semantics (briefly here, more thorough later)

Domain

A domain $\mathcal{D}$ is a set of elements which are (potentially) the *meaning* of terms and variables.

Interpretation

An interpretation $\mathcal{I}$ (over $\mathcal{D}$) assigns *meaning* to the symbols in $F_{\Sigma} \cup P_{\Sigma}$ (assigning functions to function symbols, relations to predicate symbols).

Valuation

In a given $\mathcal{D}$ and $\mathcal{I}$, a closed formula evaluates to either T or F .

Validity

A closed formula is **valid** if it evaluates to T in **all** $\mathcal{D}$ and $\mathcal{I}$.

Semantics (briefly here, more thorough later)

Domain

A domain $\mathcal{D}$ is a set of elements which are (potentially) the *meaning* of terms and variables.

Interpretation

An interpretation $\mathcal{I}$ (over $\mathcal{D}$) assigns *meaning* to the symbols in $F_{\Sigma} \cup P_{\Sigma}$ (assigning functions to function symbols, relations to predicate symbols).

Valuation

In a given $\mathcal{D}$ and $\mathcal{I}$, a closed formula evaluates to either T or F .

Validity

A closed formula is **valid** if it evaluates to T in **all** $\mathcal{D}$ and $\mathcal{I}$.

In the context of specification/verification of programs:
each $(\mathcal{D}, \mathcal{I})$ is called a '**state**'.

First-Order Semantics

From propositional to first-order semantics

- ▶ In prop. logic, an interpretation of variables with $\{T, F\}$ sufficed
- ▶ In first-order logic we must assign meaning to:
 - ▶ function symbols
 - ▶ predicate symbols
 - ▶ variables bound in quantifiers
- ▶ Respect typing: `int i`, `List l` **must** denote different items

First-Order Semantics

From propositional to first-order semantics

- ▶ In prop. logic, an interpretation of variables with $\{T, F\}$ sufficed
- ▶ In first-order logic we must assign meaning to:
 - ▶ function symbols
 - ▶ predicate symbols
 - ▶ variables bound in quantifiers
- ▶ Respect typing: `int i`, `List l` **must** denote different items

What we need (to interpret a first-order formula)

1. A **typed domain of items**
2. A mapping from **function symbols** to **functions on items**
3. A mapping from **predicate symbols** to **relation on items**
4. A mapping from **variables** to **items**

First-Order Domains

1. A **typed domain of items**:

Definition (Typed Domain)

A non-empty set $\mathcal{D}$ of items is a **domain**.

A **typing** of $\mathcal{D}$ wrt. signature Σ is a mapping $\delta : \mathcal{D} \rightarrow T_\Sigma$

First-Order Domains

1. A **typed domain of items**:

Definition (Typed Domain)

A non-empty set $\mathcal{D}$ of items is a **domain**.

A **typing** of $\mathcal{D}$ wrt. signature Σ is a mapping $\delta : \mathcal{D} \rightarrow T_\Sigma$

We require from $\mathcal{D}$ and δ that **no type is empty**:
for each $\tau \in T_\Sigma$, there is a $d \in \mathcal{D}$ with $\delta(d) = \tau$

First-Order Domains

1. A typed domain of items:

Definition (Typed Domain)

A non-empty set $\mathcal{D}$ of items is a **domain**.

A **typing** of $\mathcal{D}$ wrt. signature Σ is a mapping $\delta : \mathcal{D} \rightarrow T_\Sigma$

We require from $\mathcal{D}$ and δ that **no type is empty**:
for each $\tau \in T_\Sigma$, there is a $d \in \mathcal{D}$ with $\delta(d) = \tau$

- ▶ If $\delta(d) = \tau$, we say **d has type τ** .
- ▶ $\mathcal{D}^\tau = \{d \in \mathcal{D} \mid \delta(d) = \tau\}$ is called **subdomain of type τ** .
- ▶ It follows that $\mathcal{D}^\tau \neq \emptyset$ for each $\tau \in T_\Sigma$.

First-Order States

2. A mapping from **function symbol** to **functions on items**
3. A mapping from **predicate symbol** to **relation on items**

Definition (Interpretation, First-Order State)

Let $\mathcal{D}$ be a domain with typing δ .

First-Order States

2. A mapping from **function symbol** to **functions on items**
3. A mapping from **predicate symbol** to **relation on items**

Definition (Interpretation, First-Order State)

Let $\mathcal{D}$ be a domain with typing δ .

Let $\mathcal{I}$ be a mapping, called **interpretation**, from **function** and **predicate symbols** to **functions** and **relations on items**, respectively, such that

First-Order States

2. A mapping from **function symbol** to **functions on items**
3. A mapping from **predicate symbol** to **relation on items**

Definition (Interpretation, First-Order State)

Let $\mathcal{D}$ be a domain with typing δ .

Let $\mathcal{I}$ be a mapping, called **interpretation**, from **function** and **predicate symbols** to **functions** and **relations on items**, respectively, such that

$$\begin{aligned} \mathcal{I}(f) : \mathcal{D}^{\tau_1} \times \dots \times \mathcal{D}^{\tau_r} &\rightarrow \mathcal{D}^{\tau} && \text{when } \alpha_{\Sigma}(f) = (\tau_1, \dots, \tau_r, \tau) \\ \mathcal{I}(p) &\subseteq \mathcal{D}^{\tau_1} \times \dots \times \mathcal{D}^{\tau_r} && \text{when } \alpha_{\Sigma}(p) = (\tau_1, \dots, \tau_r) \end{aligned}$$

First-Order States

2. A mapping from **function symbol** to **functions on items**
3. A mapping from **predicate symbol** to **relation on items**

Definition (Interpretation, First-Order State)

Let $\mathcal{D}$ be a domain with typing δ .

Let $\mathcal{I}$ be a mapping, called **interpretation**, from **function** and **predicate symbols** to **functions** and **relations on items**, respectively, such that

$$\begin{aligned} \mathcal{I}(f) : \mathcal{D}^{\tau_1} \times \dots \times \mathcal{D}^{\tau_r} &\rightarrow \mathcal{D}^{\tau} && \text{when } \alpha_{\Sigma}(f) = (\tau_1, \dots, \tau_r, \tau) \\ \mathcal{I}(p) &\subseteq \mathcal{D}^{\tau_1} \times \dots \times \mathcal{D}^{\tau_r} && \text{when } \alpha_{\Sigma}(p) = (\tau_1, \dots, \tau_r) \end{aligned}$$

Then $\mathcal{S} = (\mathcal{D}, \delta, \mathcal{I})$ is a **first-order state**.

First-Order States Cont'd

Example

Signature: `int i; short j; int f(int); Object obj; <(int,int);`
 $\mathcal{D} = \{17, 2, o\}$ where all numbers are short

$$\mathcal{I}(i) = 17$$

$$\mathcal{I}(j) = 17$$

$$\mathcal{I}(\text{obj}) = o$$

$\mathcal{D}^{\text{int}}$	$\mathcal{I}(f)$
2	2
17	2

$\mathcal{D}^{\text{int}} \times \mathcal{D}^{\text{int}}$	in $\mathcal{I}(<)$?
(2, 2)	<i>F</i>
(2, 17)	<i>T</i>
(17, 2)	<i>F</i>
(17, 17)	<i>F</i>

One of uncountably many possible first-order states!

Semantics of Equality

Definition

Interpretation is fixed as $\mathcal{I}(=) = \{(d, d) \mid d \in \mathcal{D}\}$

Exercise: write down the predicate table for example domain

Signature Symbols vs. Domain Elements

- ▶ Domain elements different from the terms representing them
- ▶ First-order formulas and terms have **no access** to domain

Example

Signature: Object obj1, obj2;

Domain: $\mathcal{D} = \{o\}$

In this state, necessarily $\mathcal{I}(\text{obj1}) = \mathcal{I}(\text{obj2}) = o$

Variable Assignments

4. A mapping from variables to items

Think of variable assignment as environment for storage of local variables

Definition (Variable Assignment)

A **variable assignment** β maps variables to domain elements.
It respects the variable type, i.e., if x has type τ then $\beta(x) \in \mathcal{D}^\tau$

Definition (Modified Variable Assignment)

Let y be variable of type τ , β variable assignment, $d \in \mathcal{D}^\tau$:

$$\beta_y^d(x) := \begin{cases} \beta(x) & x \neq y \\ d & x = y \end{cases}$$

Semantic Evaluation of Terms

Given a first-order state $\mathcal{S}$ and a variable assignment β it is possible to evaluate first-order terms under $\mathcal{S}$ and β

Definition (Valuation of Terms)

$val_{\mathcal{S},\beta} : \text{Term} \rightarrow \mathcal{D}$ such that $val_{\mathcal{S},\beta}(t) \in \mathcal{D}^\tau$ for $t \in \text{Term}_\tau$:

- ▶ $val_{\mathcal{S},\beta}(x) = \beta(x)$
- ▶ $val_{\mathcal{S},\beta}(f(t_1, \dots, t_r)) = \mathcal{I}(f)(val_{\mathcal{S},\beta}(t_1), \dots, val_{\mathcal{S},\beta}(t_r))$

Semantic Evaluation of Terms Cont'd

Example

Signature: `int i; short j; int f(int);`

$\mathcal{D} = \{17, 2, o\}$ where all numbers are short

Variables: Object `obj`; `int x`;

$$\mathcal{I}(i) = 17$$

$$\mathcal{I}(j) = 17$$

$\mathcal{D}^{\text{int}}$	$\mathcal{I}(f)$
2	17
17	2

Var	β
obj	o
x	17

► $val_{\mathcal{S},\beta}(f(f(i)))$?

► $val_{\mathcal{S},\beta}(x)$?

Definition (Valuation of Formulas)

$val_{\mathcal{S},\beta}(\phi)$ for $\phi \in For$

- ▶ $val_{\mathcal{S},\beta}(p(t_1, \dots, t_r)) = T$ iff $(val_{\mathcal{S},\beta}(t_1), \dots, val_{\mathcal{S},\beta}(t_r)) \in \mathcal{I}(p)$

Definition (Valuation of Formulas)

$val_{\mathcal{S},\beta}(\phi)$ for $\phi \in For$

- ▶ $val_{\mathcal{S},\beta}(p(t_1, \dots, t_r)) = T$ iff $(val_{\mathcal{S},\beta}(t_1), \dots, val_{\mathcal{S},\beta}(t_r)) \in \mathcal{I}(p)$
- ▶ $val_{\mathcal{S},\beta}(\phi \wedge \psi) = T$ iff $val_{\mathcal{S},\beta}(\phi) = T$ and $val_{\mathcal{S},\beta}(\psi) = T$

Definition (Valuation of Formulas)

$val_{\mathcal{S},\beta}(\phi)$ for $\phi \in For$

- ▶ $val_{\mathcal{S},\beta}(p(t_1, \dots, t_r)) = T$ iff $(val_{\mathcal{S},\beta}(t_1), \dots, val_{\mathcal{S},\beta}(t_r)) \in \mathcal{I}(p)$
- ▶ $val_{\mathcal{S},\beta}(\phi \wedge \psi) = T$ iff $val_{\mathcal{S},\beta}(\phi) = T$ and $val_{\mathcal{S},\beta}(\psi) = T$
- ▶ $\neg, \vee, \rightarrow, \leftrightarrow$ as in propositional logic

Definition (Valuation of Formulas)

$val_{\mathcal{S},\beta}(\phi)$ for $\phi \in For$

- ▶ $val_{\mathcal{S},\beta}(p(t_1, \dots, t_r)) = T$ iff $(val_{\mathcal{S},\beta}(t_1), \dots, val_{\mathcal{S},\beta}(t_r)) \in \mathcal{I}(p)$
- ▶ $val_{\mathcal{S},\beta}(\phi \wedge \psi) = T$ iff $val_{\mathcal{S},\beta}(\phi) = T$ and $val_{\mathcal{S},\beta}(\psi) = T$
- ▶ $\neg, \vee, \rightarrow, \leftrightarrow$ as in propositional logic
- ▶ $val_{\mathcal{S},\beta}(\forall \tau x; \phi) = T$ iff $val_{\mathcal{S},\beta_x^d}(\phi) = T$ for all $d \in \mathcal{D}^\tau$

Semantic Evaluation of Formulas

Definition (Valuation of Formulas)

$val_{\mathcal{S},\beta}(\phi)$ for $\phi \in For$

- ▶ $val_{\mathcal{S},\beta}(p(t_1, \dots, t_r)) = T$ iff $(val_{\mathcal{S},\beta}(t_1), \dots, val_{\mathcal{S},\beta}(t_r)) \in \mathcal{I}(p)$
- ▶ $val_{\mathcal{S},\beta}(\phi \wedge \psi) = T$ iff $val_{\mathcal{S},\beta}(\phi) = T$ and $val_{\mathcal{S},\beta}(\psi) = T$
- ▶ $\neg, \vee, \rightarrow, \leftrightarrow$ as in propositional logic
- ▶ $val_{\mathcal{S},\beta}(\forall \tau x; \phi) = T$ iff $val_{\mathcal{S},\beta_x^d}(\phi) = T$ for all $d \in \mathcal{D}^\tau$
- ▶ $val_{\mathcal{S},\beta}(\exists \tau x; \phi) = T$ iff $val_{\mathcal{S},\beta_x^d}(\phi) = T$ for at least one $d \in \mathcal{D}^\tau$

Semantic Evaluation of Formulas Cont'd

Example

Signature: `short j; int f(int); Object obj; <(int,int);`

$\mathcal{D} = \{17, 2, o\}$ where all numbers are short

$\mathcal{I}(j) = 17$ $\mathcal{I}(\text{obj}) = o$			
$\mathcal{D}^{\text{int}}$	$\mathcal{I}(f)$	$\mathcal{D}^{\text{int}} \times \mathcal{D}^{\text{int}}$	in $\mathcal{I}(<)$?
2	2	(2, 2)	F
17	2	(2, 17)	T
		(17, 2)	F
		(17, 17)	F

- ▶ $\text{val}_{\mathcal{S},\beta}(f(j) < j) ?$
- ▶ $\text{val}_{\mathcal{S},\beta}(\exists \text{int } x; f(x) = x) ?$
- ▶ $\text{val}_{\mathcal{S},\beta}(\forall \text{Object } o1; \forall \text{Object } o2; o1 = o2) ?$

Semantic Notions

Definition (Satisfiability, Truth, Validity)

$$\begin{array}{lll} \text{val}_{\mathcal{S},\beta}(\phi) = T & & (\phi \text{ is } \mathbf{satisfiable}) \\ \mathcal{S} \models \phi & \text{iff for all } \beta : \text{val}_{\mathcal{S},\beta}(\phi) = T & (\phi \text{ is } \mathbf{true} \text{ in } \mathcal{S}) \\ \models \phi & \text{iff for all } \mathcal{S} : \mathcal{S} \models \phi & (\phi \text{ is } \mathbf{valid}) \end{array}$$

Closed formulas that are satisfiable are also true: one top-level notion

Definition (Satisfiability, Truth, Validity)

$$\begin{array}{lll} \text{val}_{\mathcal{S},\beta}(\phi) = T & & (\phi \text{ is } \mathbf{satisfiable}) \\ \mathcal{S} \models \phi & \text{iff for all } \beta : \text{val}_{\mathcal{S},\beta}(\phi) = T & (\phi \text{ is } \mathbf{true} \text{ in } \mathcal{S}) \\ \models \phi & \text{iff for all } \mathcal{S} : \mathcal{S} \models \phi & (\phi \text{ is } \mathbf{valid}) \end{array}$$

Closed formulas that are satisfiable are also true: one top-level notion

Example

- ▶ $f(j) < j$ is true in $\mathcal{S}$
- ▶ $\exists \text{int } x; i = x$ is valid
- ▶ $\exists \text{int } x; \neg(x = x)$ is not satisfiable

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

► $\neg(\phi \wedge \psi) \leftrightarrow$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

► $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

► $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

► $\neg(\phi \vee \psi) \leftrightarrow$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

► $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

► $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

- ▶ $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$
- ▶ $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$
- ▶ $(\text{true} \wedge \phi) \leftrightarrow$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

- ▶ $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$
- ▶ $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$
- ▶ $(\text{true} \wedge \phi) \leftrightarrow \phi$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

► $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

► $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

► $(\text{true} \wedge \phi) \leftrightarrow \phi$

► $(\text{false} \vee \phi) \leftrightarrow \phi$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

► $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

► $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

► $(\text{true} \wedge \phi) \leftrightarrow \phi$

► $(\text{false} \vee \phi) \leftrightarrow \phi$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

► $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

► $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

► $(\text{true} \wedge \phi) \leftrightarrow \phi$

► $(\text{false} \vee \phi) \leftrightarrow \phi$

► $\text{true} \vee \phi$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

► $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

► $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

► $(\text{true} \wedge \phi) \leftrightarrow \phi$

► $(\text{false} \vee \phi) \leftrightarrow \phi$

► $\text{true} \vee \phi$

► $\neg(\text{false} \wedge \phi)$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

► $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

► $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

► $(\text{true} \wedge \phi) \leftrightarrow \phi$

► $(\text{false} \vee \phi) \leftrightarrow \phi$

► $\text{true} \vee \phi$

► $\neg(\text{false} \wedge \phi)$

► $(\phi \rightarrow \psi) \leftrightarrow$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

▶ $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

▶ $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

▶ $(\text{true} \wedge \phi) \leftrightarrow \phi$

▶ $(\text{false} \vee \phi) \leftrightarrow \phi$

▶ $\text{true} \vee \phi$

▶ $\neg(\text{false} \wedge \phi)$

▶ $(\phi \rightarrow \psi) \leftrightarrow (\neg\phi \vee \psi)$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

► $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

► $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

► $(\text{true} \wedge \phi) \leftrightarrow \phi$

► $(\text{false} \vee \phi) \leftrightarrow \phi$

► $\text{true} \vee \phi$

► $\neg(\text{false} \wedge \phi)$

► $(\phi \rightarrow \psi) \leftrightarrow (\neg\phi \vee \psi)$

► $\phi \rightarrow \text{true}$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

► $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

► $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

► $(\text{true} \wedge \phi) \leftrightarrow \phi$

► $(\text{false} \vee \phi) \leftrightarrow \phi$

► $\text{true} \vee \phi$

► $\neg(\text{false} \wedge \phi)$

► $(\phi \rightarrow \psi) \leftrightarrow (\neg\phi \vee \psi)$

► $\phi \rightarrow \text{true}$

► $\text{false} \rightarrow \phi$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

► $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

► $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

► $(\text{true} \wedge \phi) \leftrightarrow \phi$

► $(\text{false} \vee \phi) \leftrightarrow \phi$

► $\text{true} \vee \phi$

► $\neg(\text{false} \wedge \phi)$

► $(\phi \rightarrow \psi) \leftrightarrow (\neg\phi \vee \psi)$

► $\phi \rightarrow \text{true}$

► $\text{false} \rightarrow \phi$

► $(\text{true} \rightarrow \phi) \leftrightarrow$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

► $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

► $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

► $(\text{true} \wedge \phi) \leftrightarrow \phi$

► $(\text{false} \vee \phi) \leftrightarrow \phi$

► $\text{true} \vee \phi$

► $\neg(\text{false} \wedge \phi)$

► $(\phi \rightarrow \psi) \leftrightarrow (\neg\phi \vee \psi)$

► $\phi \rightarrow \text{true}$

► $\text{false} \rightarrow \phi$

► $(\text{true} \rightarrow \phi) \leftrightarrow \phi$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

► $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

► $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

► $(\text{true} \wedge \phi) \leftrightarrow \phi$

► $(\text{false} \vee \phi) \leftrightarrow \phi$

► $\text{true} \vee \phi$

► $\neg(\text{false} \wedge \phi)$

► $(\phi \rightarrow \psi) \leftrightarrow (\neg\phi \vee \psi)$

► $\phi \rightarrow \text{true}$

► $\text{false} \rightarrow \phi$

► $(\text{true} \rightarrow \phi) \leftrightarrow \phi$

► $(\phi \rightarrow \text{false}) \leftrightarrow$

Useful Valid Formulas

Let ϕ and ψ be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

► $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

► $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

► $(\text{true} \wedge \phi) \leftrightarrow \phi$

► $(\text{false} \vee \phi) \leftrightarrow \phi$

► $\text{true} \vee \phi$

► $\neg(\text{false} \wedge \phi)$

► $(\phi \rightarrow \psi) \leftrightarrow (\neg\phi \vee \psi)$

► $\phi \rightarrow \text{true}$

► $\text{false} \rightarrow \phi$

► $(\text{true} \rightarrow \phi) \leftrightarrow \phi$

► $(\phi \rightarrow \text{false}) \leftrightarrow \neg\phi$

Useful Valid Formulas

Assume that x is the only variable which may appear freely in ϕ or ψ .

The following formulas are valid:

Useful Valid Formulas

Assume that x is the only variable which may appear freely in ϕ or ψ .

The following formulas are valid:

► $\neg(\exists \tau x; \phi) \leftrightarrow$

Useful Valid Formulas

Assume that x is the only variable which may appear freely in ϕ or ψ .

The following formulas are valid:

$$\blacktriangleright \neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$$

Useful Valid Formulas

Assume that x is the only variable which may appear freely in ϕ or ψ .

The following formulas are valid:

- ▶ $\neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$
- ▶ $\neg(\forall \tau x; \phi) \leftrightarrow$

Useful Valid Formulas

Assume that x is the only variable which may appear freely in ϕ or ψ .

The following formulas are valid:

- ▶ $\neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$
- ▶ $\neg(\forall \tau x; \phi) \leftrightarrow \exists \tau x; \neg\phi$

Useful Valid Formulas

Assume that x is the only variable which may appear freely in ϕ or ψ .

The following formulas are valid:

- ▶ $\neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$
- ▶ $\neg(\forall \tau x; \phi) \leftrightarrow \exists \tau x; \neg\phi$
- ▶ $(\forall \tau x; (\phi \wedge \psi)) \leftrightarrow$

Useful Valid Formulas

Assume that x is the only variable which may appear freely in ϕ or ψ .

The following formulas are valid:

- ▶ $\neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$
- ▶ $\neg(\forall \tau x; \phi) \leftrightarrow \exists \tau x; \neg\phi$
- ▶ $(\forall \tau x; (\phi \wedge \psi)) \leftrightarrow (\forall \tau x; \phi) \wedge (\forall \tau x; \psi)$

Useful Valid Formulas

Assume that x is the only variable which may appear freely in ϕ or ψ .

The following formulas are valid:

- ▶ $\neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$
- ▶ $\neg(\forall \tau x; \phi) \leftrightarrow \exists \tau x; \neg\phi$
- ▶ $(\forall \tau x; (\phi \wedge \psi)) \leftrightarrow (\forall \tau x; \phi) \wedge (\forall \tau x; \psi)$
- ▶ $(\exists \tau x; (\phi \vee \psi)) \leftrightarrow$

Useful Valid Formulas

Assume that x is the only variable which may appear freely in ϕ or ψ .

The following formulas are valid:

- ▶ $\neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$
- ▶ $\neg(\forall \tau x; \phi) \leftrightarrow \exists \tau x; \neg\phi$
- ▶ $(\forall \tau x; (\phi \wedge \psi)) \leftrightarrow (\forall \tau x; \phi) \wedge (\forall \tau x; \psi)$
- ▶ $(\exists \tau x; (\phi \vee \psi)) \leftrightarrow (\exists \tau x; \phi) \vee (\exists \tau x; \psi)$

Useful Valid Formulas

Assume that x is the only variable which may appear freely in ϕ or ψ .

The following formulas are valid:

- ▶ $\neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$
- ▶ $\neg(\forall \tau x; \phi) \leftrightarrow \exists \tau x; \neg\phi$
- ▶ $(\forall \tau x; (\phi \wedge \psi)) \leftrightarrow (\forall \tau x; \phi) \wedge (\forall \tau x; \psi)$
- ▶ $(\exists \tau x; (\phi \vee \psi)) \leftrightarrow (\exists \tau x; \phi) \vee (\exists \tau x; \psi)$

Which of the following formulas are valid?

1. $(\forall \tau x; (\phi \vee \psi)) \leftrightarrow (\forall \tau x; \phi) \vee (\forall \tau x; \psi)$
2. $(\exists \tau x; (\phi \wedge \psi)) \leftrightarrow (\exists \tau x; \phi) \wedge (\exists \tau x; \psi)$

Useful Valid Formulas

Assume that x is the only variable which may appear freely in ϕ or ψ .

The following formulas are valid:

- ▶ $\neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$
- ▶ $\neg(\forall \tau x; \phi) \leftrightarrow \exists \tau x; \neg\phi$
- ▶ $(\forall \tau x; (\phi \wedge \psi)) \leftrightarrow (\forall \tau x; \phi) \wedge (\forall \tau x; \psi)$
- ▶ $(\exists \tau x; (\phi \vee \psi)) \leftrightarrow (\exists \tau x; \phi) \vee (\exists \tau x; \psi)$

Which of the following formulas are valid?

1. $(\forall \tau x; (\phi \vee \psi)) \leftrightarrow (\forall \tau x; \phi) \vee (\forall \tau x; \psi)$
2. $(\exists \tau x; (\phi \wedge \psi)) \leftrightarrow (\exists \tau x; \phi) \wedge (\exists \tau x; \psi)$

Poll

Remark on Concrete Syntax

	Text book	SPIN	KeY
Negation	$\neg$	!	!
Conjunction	$\wedge$	&&	&
Disjunction	$\vee$		
Implication	$\rightarrow, \supset$	$\rightarrow$	$\rightarrow$
Equivalence	$\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$
Universal Quantifier	$\forall x; \phi$	n/a	<code>\forall x; \phi</code>
Existential Quantifier	$\exists x; \phi$	n/a	<code>\exists x; \phi</code>
Value equality	=	==	=

Reasoning by Syntactic Transformation

Prove validity of ϕ by **syntactic** transformation of ϕ

Reasoning by Syntactic Transformation

Prove validity of ϕ by **syntactic** transformation of ϕ

Sequent Calculus based on notion of **sequent**:

$$\underbrace{\psi_1, \dots, \psi_m}_{\text{antecedent}} \Rightarrow \underbrace{\phi_1, \dots, \phi_n}_{\text{succedent}}$$

Reasoning by Syntactic Transformation

Prove validity of ϕ by **syntactic** transformation of ϕ

Sequent Calculus based on notion of **sequent**:

$$\underbrace{\psi_1, \dots, \psi_m}_{\text{antecedent}} \Rightarrow \underbrace{\phi_1, \dots, \phi_n}_{\text{succedent}}$$

has same meaning as

$$(\psi_1 \wedge \dots \wedge \psi_m) \rightarrow (\phi_1 \vee \dots \vee \phi_n)$$

Reasoning by Syntactic Transformation

Prove validity of ϕ by **syntactic** transformation of ϕ

Sequent Calculus based on notion of **sequent**:

$$\underbrace{\psi_1, \dots, \psi_m}_{\text{antecedent}} \Rightarrow \underbrace{\phi_1, \dots, \phi_n}_{\text{succedent}}$$

has same meaning as

$$(\psi_1 \wedge \dots \wedge \psi_m) \rightarrow (\phi_1 \vee \dots \vee \phi_n)$$

which (for closed formulas ψ_i, ϕ_i) is equivalent to

$$\{\psi_1, \dots, \psi_m\} \models \phi_1 \vee \dots \vee \phi_n$$

Notation for Sequents

$$\psi_1, \dots, \psi_m \Rightarrow \phi_1, \dots, \phi_n$$

Consider antecedent/succedent as sets of formulas (possibly empty)

Notation for Sequents

$$\psi_1, \dots, \psi_m \Rightarrow \phi_1, \dots, \phi_n$$

Consider antecedent/succedent as sets of formulas (possibly empty)

Schema Variables

$\phi, \psi, \dots$ match formulas.

$\Gamma, \Delta, \dots$ match sets of formulas.

Characterize infinitely many sequents with single schematic sequent, e.g.,

$$\Gamma \Rightarrow \phi \wedge \psi, \Delta$$

matches any sequent with occurrence of conjunction in succedent.

Here, we call $\phi \wedge \psi$ **main formula** and Γ, Δ **side formulas** of sequent

Sequent Calculus Rules

Write syntactic transformation schema for sequents,
reflecting semantics of connectives

$$\text{RuleName} \frac{\overbrace{\Gamma_1 \Rightarrow \Delta_1 \quad \dots \quad \Gamma_r \Rightarrow \Delta_r}^{\text{premisses}}}{\underbrace{\Gamma \Rightarrow \Delta}_{\text{conclusion}}}$$

Meaning: For proving the conclusion, it suffices to prove all premisses.

Sequent Calculus Rules

Write syntactic transformation schema for sequents,
reflecting semantics of connectives

$$\text{RuleName} \frac{\overbrace{\Gamma_1 \Rightarrow \Delta_1 \quad \dots \quad \Gamma_r \Rightarrow \Delta_r}^{\text{premises}}}{\underbrace{\Gamma \Rightarrow \Delta}_{\text{conclusion}}}$$

Meaning: For proving the conclusion, it suffices to prove all premisses.

Example

$$\text{andRight} \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$$

Sequent Calculus Rules

Write syntactic transformation schema for sequents,
reflecting semantics of connectives

$$\text{RuleName} \frac{\overbrace{\Gamma_1 \Rightarrow \Delta_1 \quad \dots \quad \Gamma_r \Rightarrow \Delta_r}^{\text{premises}}}{\underbrace{\Gamma \Rightarrow \Delta}_{\text{conclusion}}}$$

Meaning: For proving the conclusion, it suffices to prove all premisses.

Example

$$\text{andRight} \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$$

Admissible to have no premisses (then the rule is called 'axiom').

A rule is **sound** (correct) iff the validity of all premisses implies the validity of the conclusion.

'Propositional' Sequent Calculus Rules

$$\text{close} \quad \frac{}{\Gamma, \phi \Rightarrow \phi, \Delta}$$

'Propositional' Sequent Calculus Rules

$$\text{close} \quad \frac{}{\Gamma, \phi \Rightarrow \phi, \Delta} \quad \text{true} \quad \frac{}{\Gamma \Rightarrow \text{true}, \Delta}$$

'Propositional' Sequent Calculus Rules

$$\text{close} \quad \frac{}{\Gamma, \phi \Rightarrow \phi, \Delta} \quad \text{true} \quad \frac{}{\Gamma \Rightarrow \text{true}, \Delta} \quad \text{false} \quad \frac{}{\Gamma, \text{false} \Rightarrow \Delta}$$

'Propositional' Sequent Calculus Rules

$$\text{close} \quad \frac{}{\Gamma, \phi \Rightarrow \phi, \Delta} \quad \text{true} \quad \frac{}{\Gamma \Rightarrow \text{true}, \Delta} \quad \text{false} \quad \frac{}{\Gamma, \text{false} \Rightarrow \Delta}$$

	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg \phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg \phi, \Delta}$

'Propositional' Sequent Calculus Rules

$$\text{close} \quad \frac{}{\Gamma, \phi \Rightarrow \phi, \Delta} \quad \text{true} \quad \frac{}{\Gamma \Rightarrow \text{true}, \Delta} \quad \text{false} \quad \frac{}{\Gamma, \text{false} \Rightarrow \Delta}$$

	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg \phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg \phi, \Delta}$
and	$\frac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$

'Propositional' Sequent Calculus Rules

$$\text{close} \quad \frac{}{\Gamma, \phi \Rightarrow \phi, \Delta} \quad \text{true} \quad \frac{}{\Gamma \Rightarrow \text{true}, \Delta} \quad \text{false} \quad \frac{}{\Gamma, \text{false} \Rightarrow \Delta}$$

	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg \phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg \phi, \Delta}$
and	$\frac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$
or	$\frac{\Gamma, \phi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \vee \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}$

'Propositional' Sequent Calculus Rules

$$\text{close} \quad \frac{}{\Gamma, \phi \Rightarrow \phi, \Delta} \quad \text{true} \quad \frac{}{\Gamma \Rightarrow \text{true}, \Delta} \quad \text{false} \quad \frac{}{\Gamma, \text{false} \Rightarrow \Delta}$$

	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg \phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg \phi, \Delta}$
and	$\frac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$
or	$\frac{\Gamma, \phi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \vee \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}$
imp	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \rightarrow \psi \Rightarrow \Delta}$	

'Propositional' Sequent Calculus Rules

$$\text{close} \quad \frac{}{\Gamma, \phi \Rightarrow \phi, \Delta} \quad \text{true} \quad \frac{}{\Gamma \Rightarrow \text{true}, \Delta} \quad \text{false} \quad \frac{}{\Gamma, \text{false} \Rightarrow \Delta}$$

	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg \phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg \phi, \Delta}$
and	$\frac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$
or	$\frac{\Gamma, \phi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \vee \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}$
imp	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \rightarrow \psi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \rightarrow \psi, \Delta}$

Sequent Calculus Proofs

Goal to prove: $\mathcal{G} = \psi_1, \dots, \psi_m \Rightarrow \phi_1, \dots, \phi_n$

- ▶ find rule $\mathcal{R}$ whose conclusion **matches** $\mathcal{G}$
- ▶ instantiate $\mathcal{R}$ such that its conclusion is **identical** to $\mathcal{G}$
- ▶ apply that instantiation to all premisses of $\mathcal{R}$, resulting in new goals $\mathcal{G}_1, \dots, \mathcal{G}_r$
- ▶ recursively find proofs for $\mathcal{G}_1, \dots, \mathcal{G}_r$
- ▶ tree structure with goal as root
- ▶ **close** proof branch when applying rule without premiss

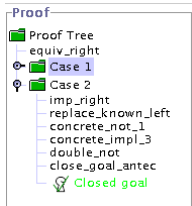
Sequent Calculus Proofs

Goal to prove: $\mathcal{G} = \psi_1, \dots, \psi_m \Rightarrow \phi_1, \dots, \phi_n$

- ▶ find rule $\mathcal{R}$ whose conclusion **matches** $\mathcal{G}$
- ▶ instantiate $\mathcal{R}$ such that its conclusion is **identical** to $\mathcal{G}$
- ▶ apply that instantiation to all premisses of $\mathcal{R}$, resulting in new goals $\mathcal{G}_1, \dots, \mathcal{G}_r$
- ▶ recursively find proofs for $\mathcal{G}_1, \dots, \mathcal{G}_r$
- ▶ tree structure with goal as root
- ▶ **close** proof branch when applying rule without premiss

Goal-directed proof search

- ▶ Paper proofs: root at bottom, grow upwards
- ▶ KeY tool proofs: root at top, grow downwards



A Simple Proof

$$\frac{\frac{}{} \quad \frac{}{} }{} \quad \frac{}{} }{} \Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q$$

A Simple Proof

$$\frac{\frac{}{p \wedge (p \rightarrow q) \Rightarrow q}}{\Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q}$$

A Simple Proof

$$\frac{\frac{\frac{}{p, (p \rightarrow q) \Rightarrow q}}{p \wedge (p \rightarrow q) \Rightarrow q}}{\Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q}$$

A Simple Proof

$$\frac{\frac{\frac{}{p \Rightarrow p, q}}{p, (p \rightarrow q) \Rightarrow q}}{p \wedge (p \rightarrow q) \Rightarrow q}}{\Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q}$$

A Simple Proof

$$\begin{array}{c} \text{CLOSE} \frac{*}{p \Rightarrow p, q} \quad \frac{*}{p, q \Rightarrow q} \text{CLOSE} \\ \hline p, (p \rightarrow q) \Rightarrow q \\ \hline p \wedge (p \rightarrow q) \Rightarrow q \\ \hline \Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q \end{array}$$

A Simple Proof

$$\frac{\frac{\text{CLOSE} \quad *}{p \Rightarrow p, q} \quad \frac{*}{p, q \Rightarrow q} \text{CLOSE}}{p, (p \rightarrow q) \Rightarrow q} \\ \frac{p \wedge (p \rightarrow q) \Rightarrow q}{\Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q}$$

A proof is **closed** iff all its branches are closed

Demo

prop.key

Proving Validity of First-Order Formulas

Proving a universally quantified formula

Claim: $\forall x; \phi$ is true

How is such a claim proved in Mathematics?

All even numbers are divisible by 2 $\forall \text{int } x; (\text{even}(x) \rightarrow \text{divByTwo}(x))$

Let c be an arbitrary number Declare “unused” constant `int c`

The even number c is divisible by 2 Prove $\text{even}(c) \rightarrow \text{divByTwo}(c)$

Proving Validity of First-Order Formulas

Proving a universally quantified formula

Claim: $\forall \tau x; \phi$ is true

How is such a claim proved in Mathematics?

All even numbers are divisible by 2 $\forall \text{int } x; (\text{even}(x) \rightarrow \text{divByTwo}(x))$

Let c be an arbitrary number Declare “unused” constant `int c`

The even number c is divisible by 2 Prove $\text{even}(c) \rightarrow \text{divByTwo}(c)$

Sequent rule $\forall$ -right

$$\text{forallRight} \frac{\Gamma \Rightarrow [x/c] \phi, \Delta}{\Gamma \Rightarrow \forall \tau x; \phi, \Delta}$$

- ▶ $[x/c] \phi$ is result of replacing each occurrence of x in ϕ with c
- ▶ c **new** constant of type τ

Proving Validity of First-Order Formulas Cont'd

Proving an existentially quantified formula

Claim: $\exists x; \phi$ is true

How is such a claim proved in Mathematics?

There is at least one prime number $\exists \text{int } x; \text{prime}(x)$

Provide any “witness”, say, 7 Use variable-free term `int 7`

7 is a prime number Prove `prime(7)`

Proving Validity of First-Order Formulas Cont'd

Proving an existentially quantified formula

Claim: $\exists \tau x; \phi$ is true

How is such a claim proved in Mathematics?

There is at least one prime number $\exists \text{int } x; \text{prime}(x)$

Provide any “witness”, say, 7 Use variable-free term $\text{int } 7$

7 is a prime number Prove $\text{prime}(7)$

Sequent rule $\exists$ -right

$$\text{existsRight} \frac{\Gamma \Rightarrow [x/t] \phi, \exists \tau x; \phi, \Delta}{\Gamma \Rightarrow \exists \tau x; \phi, \Delta}$$

- ▶ t any variable-free term of type τ
- ▶ We might need other instances besides t ! Keep $\exists \tau x; \phi$

Proving Validity of First-Order Formulas Cont'd

Using a universally quantified formula

We assume $\forall x; \phi$ is true.

How is such a fact **used** in a **Mathematical proof**?

We know that all primes are odd $\forall \text{int } x; (\text{prime}(x) \rightarrow \text{odd}(x))$

In particular, this holds for 17 Use variable-free term `int 17`

We know: if 17 is prime it is odd $\text{prime}(17) \rightarrow \text{odd}(17)$

Proving Validity of First-Order Formulas Cont'd

Using a universally quantified formula

We assume $\forall \tau x; \phi$ is true.

How is such a fact **used** in a **Mathematical proof**?

We know that all primes are odd $\forall \text{int } x; (\text{prime}(x) \rightarrow \text{odd}(x))$

In particular, this holds for 17 Use variable-free term `int 17`

We know: if 17 is prime it is odd $\text{prime}(17) \rightarrow \text{odd}(17)$

Sequent rule $\forall$ -left

$$\text{forallLeft} \frac{\Gamma, \forall \tau x; \phi, [x/t] \phi \Rightarrow \Delta}{\Gamma, \forall \tau x; \phi \Rightarrow \Delta}$$

- ▶ t any variable-free term of type τ
- ▶ We might need other instances besides t ! Keep $\forall \tau x; \phi$

Proving Validity of First-Order Formulas Cont'd

Using an existentially quantified formula

We assume $\exists x; \phi$ is true

How is such a fact **used** in a **Mathematical proof**?

We know such an element exists. Let's give that element it a new name.

Proving Validity of First-Order Formulas Cont'd

Using an existentially quantified formula

We assume $\exists \tau x; \phi$ is true

How is such a fact **used** in a **Mathematical proof**?

We know such an element exists. Let's give that element it a new name.

Sequent rule $\exists$ -left

$$\text{existsLeft} \frac{\Gamma, [x/c] \phi \Rightarrow \Delta}{\Gamma, \exists \tau x; \phi \Rightarrow \Delta}$$

► c **new** constant of type τ

Proving Validity of First-Order Formulas Cont'd

Using an equation between terms

We assume $t = t'$ is true

How is such a fact used in a Mathematical proof?

$$x = y-1 \Rightarrow 1 = x+1/y$$

Use $x = y-1$ to modify $x+1/y$:

Replace x in succedent with right-hand side of antecedent

Proving Validity of First-Order Formulas Cont'd

Using an equation between terms

We assume $t = t'$ is true

How is such a fact used in a Mathematical proof?

$$x = y-1 \Rightarrow 1 = x+1/y$$

Use $x = y-1$ to modify $x+1/y$:

Replace x in succedent with right-hand side of antecedent

$$x = y-1 \Rightarrow 1 = y-1+1/y$$

Proving Validity of First-Order Formulas Cont'd

Using an equation between terms

We assume $t = t'$ is true

How is such a fact used in a Mathematical proof?

$$x = y-1 \Rightarrow 1 = x+1/y$$

Use $x = y-1$ to modify $x+1/y$:

Replace x in succedent with right-hand side of antecedent

$$x = y-1 \Rightarrow 1 = y-1+1/y$$

Sequent rule =-left

$$\text{applyEqL} \quad \frac{\Gamma, t = t', [t/t'] \phi \Rightarrow \Delta}{\Gamma, t = t', \phi \Rightarrow \Delta} \quad \text{applyEqR} \quad \frac{\Gamma, t = t' \Rightarrow [t/t'] \phi, \Delta}{\Gamma, t = t' \Rightarrow \phi, \Delta}$$

- ▶ Always replace left- with right-hand side (use **eqSymm** if necessary)
- ▶ t, t' variable-free terms of the same type

Proving Validity of First-Order Formulas Cont'd

Closing a subgoal in a proof

- ▶ We derived a sequent that is trivially valid

$$\text{close} \frac{}{\Gamma, \phi \Rightarrow \phi, \Delta} \quad \text{true} \frac{}{\Gamma \Rightarrow \text{true}, \Delta} \quad \text{false} \frac{}{\Gamma, \text{false} \Rightarrow \Delta}$$

- ▶ We derived an **equation** that is trivially valid

$$\text{eqClose} \frac{}{\Gamma \Rightarrow t = t, \Delta}$$

Sequent Calculus for FOL at One Glance

	left side, antecedent	right side, succedent
$\forall$	$\frac{\Gamma, \forall \tau x; \phi, [x/t'] \phi \Rightarrow \Delta}{\Gamma, \forall \tau x; \phi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow [x/c] \phi, \Delta}{\Gamma \Rightarrow \forall \tau x; \phi, \Delta}$
$\exists$	$\frac{\Gamma, [x/c] \phi \Rightarrow \Delta}{\Gamma, \exists \tau x; \phi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow [x/t'] \phi, \exists \tau x; \phi, \Delta}{\Gamma \Rightarrow \exists \tau x; \phi, \Delta}$
$=$	$\frac{\Gamma, t = t' \Rightarrow [t/t'] \phi, \Delta}{\Gamma, t = t' \Rightarrow \phi, \Delta}$ (+ application rule on left side)	$\frac{}{\Gamma \Rightarrow t = t, \Delta}$

- ▶ $[t/t'] \phi$ is result of replacing each occurrence of t in ϕ with t'
- ▶ t, t' variable-free terms of type τ
- ▶ c **new** constant of type τ (occurs not on current proof branch)
- ▶ Equations can be reversed by commutativity

Recap: 'Propositional' Sequent Calculus Rules

main	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg \phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg \phi, \Delta}$
and	$\frac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$
or	$\frac{\Gamma, \phi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \vee \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}$
imp	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \rightarrow \psi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \rightarrow \psi, \Delta}$
close	$\frac{}{\Gamma, \phi \Rightarrow \phi, \Delta}$	true $\frac{}{\Gamma \Rightarrow \text{true}, \Delta}$ false $\frac{}{\Gamma, \text{false} \Rightarrow \Delta}$

Proving Validity of First-Order Formulas Cont'd

Example (A simple theorem about binary relations)

$$\exists x; \forall y; p(x, y) \implies \forall y; \exists x; p(x, y)$$

“Untyped” logic: let type of x and y be *any*

Proving Validity of First-Order Formulas Cont'd

Example (A simple theorem about binary relations)

$$\frac{\forall y; p(\textcolor{red}{c}, y) \Rightarrow \forall y; \exists x; p(x, y)}{\exists x; \forall y; p(x, y) \Rightarrow \forall y; \exists x; p(x, y)}$$

$\exists$ -left: substitute **new** constant c for x

Proving Validity of First-Order Formulas Cont'd

Example (A simple theorem about binary relations)

$$\frac{\frac{\forall y; p(c, y) \Rightarrow \exists x; p(x, d)}{\forall y; p(\textcolor{red}{c}, y) \Rightarrow \forall y; \exists x; p(x, y)}}{\exists x; \forall y; p(x, y) \Rightarrow \forall y; \exists x; p(x, y)}$$

$\forall$ -right: substitute **new** constant d for y

Proving Validity of First-Order Formulas Cont'd

Example (A simple theorem about binary relations)

$$\frac{\frac{\frac{p(c, d), \forall y; p(c, y) \Rightarrow \exists x; p(x, d)}{\forall y; p(c, y) \Rightarrow \exists x; p(x, d)}}{\forall y; p(c, y) \Rightarrow \forall y; \exists x; p(x, y)}}{\exists x; \forall y; p(x, y) \Rightarrow \forall y; \exists x; p(x, y)}$$

$\forall$ -left: free to substitute **arbitrary** term (of right type) for y , choose d

Proving Validity of First-Order Formulas Cont'd

Example (A simple theorem about binary relations)

$$\frac{\frac{\frac{p(c, d), \forall y; p(c, y) \Rightarrow p(\textcolor{red}{c}, d), \exists x; p(x, d)}{p(c, \textcolor{red}{d}), \forall y; p(c, y) \Rightarrow \exists x; p(x, d)}}{\forall y; p(c, y) \Rightarrow \exists x; p(x, \textcolor{red}{d})}}{\frac{\forall y; p(\textcolor{red}{c}, y) \Rightarrow \forall y; \exists x; p(x, y)}{\exists x; \forall y; p(x, y) \Rightarrow \forall y; \exists x; p(x, y)}}$$

$\exists$ -right: free to substitute **arbitrary** term (of right type) for x , choose c

Proving Validity of First-Order Formulas Cont'd

Example (A simple theorem about binary relations)

$$\begin{array}{c} * \\ \hline p(c, d), \forall y; p(c, y) \Rightarrow p(\textcolor{red}{c}, d), \exists x; p(x, d) \\ \hline p(c, \textcolor{red}{d}), \forall y; p(c, y) \Rightarrow \exists x; p(x, d) \\ \hline \forall y; p(c, y) \Rightarrow \exists x; p(x, \textcolor{red}{d}) \\ \hline \forall y; p(\textcolor{red}{c}, y) \Rightarrow \forall y; \exists x; p(x, y) \\ \hline \exists x; \forall y; p(x, y) \Rightarrow \forall y; \exists x; p(x, y) \end{array}$$

Close

Proving Validity of First-Order Formulas Cont'd

Example (A simple theorem about binary relations)

$$\begin{array}{c} * \\ \hline p(c, d), \forall y; p(c, y) \Rightarrow p(\textcolor{red}{c}, d), \exists x; p(x, d) \\ \hline p(c, \textcolor{red}{d}), \forall y; p(c, y) \Rightarrow \exists x; p(x, d) \\ \hline \forall y; p(c, y) \Rightarrow \exists x; p(x, \textcolor{red}{d}) \\ \hline \forall y; p(\textcolor{red}{c}, y) \Rightarrow \forall y; \exists x; p(x, y) \\ \hline \exists x; \forall y; p(x, y) \Rightarrow \forall y; \exists x; p(x, y) \end{array}$$

Demo

relSimple.key

Proving Validity of First-Order Formulas Cont'd

Using an existentially quantified formula

Proving Validity of First-Order Formulas Cont'd

Using an existentially quantified formula

Let x, y denote integer constants, both are not zero.

$$\neg(x = 0), \neg(y = 0) \quad \Rightarrow$$

Proving Validity of First-Order Formulas Cont'd

Using an existentially quantified formula

Let x, y denote integer constants, both are not zero. We know further that x divides y .

$$\neg(x = 0), \neg(y = 0), \exists \text{int } k; y = k * x \Rightarrow$$

Proving Validity of First-Order Formulas Cont'd

Using an existentially quantified formula

Let x, y denote integer constants, both are not zero. We know further that x divides y .

Show: $(y/x) * x = y$ ('/' is division on integers, i.e., the equation is not always true, e.g. $y = 1, x = 2$)

$$\neg(x = 0), \neg(y = 0), \exists \text{int } k; y = k * x \implies (y/x) * x = y$$

Proving Validity of First-Order Formulas Cont'd

Using an existentially quantified formula

Let x, y denote integer constants, both are not zero. We know further that x divides y .

Show: $(y/x) * x = y$ ('/' is division on integers, i.e., the equation is not always true, e.g. $y = 1, x = 2$)

Proof: We know x divides y , i.e. there exists a k such that $y = k * x$.

Let now c denote such a k .

$$\frac{\neg(x = 0), \neg(y = 0), y = c * x \implies (y/x) * x = y}{\neg(x = 0), \neg(y = 0), \exists \text{int } k; y = k * x \implies (y/x) * x = y}$$

Proving Validity of First-Order Formulas Cont'd

Using an existentially quantified formula

Let x, y denote integer constants, both are not zero. We know further that x divides y .

Show: $(y/x) * x = y$ ('/' is division on integers, i.e., the equation is not always true, e.g. $y = 1, x = 2$)

Proof: We know x divides y , i.e. there exists a k such that $y = k * x$. Let now c denote such a k . Hence we can replace y by $c * x$ on the right side.

$$\frac{\neg(x = 0), \neg(y = 0), y = c * x \implies ((c * x)/x) * x = y}{\neg(x = 0), \neg(y = 0), y = c * x \implies (y/x) * x = y}}{\neg(x = 0), \neg(y = 0), \exists \text{int } k; y = k * x \implies (y/x) * x = y}$$

Proving Validity of First-Order Formulas Cont'd

Using an existentially quantified formula

Let x, y denote integer constants, both are not zero. We know further that x divides y .

Show: $(y/x) * x = y$ ('/' is division on integers, i.e., the equation is not always true, e.g. $y = 1, x = 2$)

Proof: We know x divides y , i.e. there exists a k such that $y = k * x$. Let now c denote such a k . Hence we can replace y by $c * x$ on the right side. ... $\square$

$$\begin{array}{c} * \\ \hline \vdots \\ \hline \neg(x = 0), \neg(y = 0), y = c * x \implies ((c * x)/x) * x = y \\ \hline \neg(x = 0), \neg(y = 0), y = c * x \implies (y/x) * x = y \\ \hline \neg(x = 0), \neg(y = 0), \exists \text{int } k; y = k * x \implies (y/x) * x = y \end{array}$$

Features of the KeY Theorem Prover

Demo

`rel.key, twoInstances.key`

Feature List

- ▶ Can work on multiple proofs simultaneously (task list)
- ▶ Point-and-click navigation within proof
- ▶ Undo proof steps, prune proof trees
- ▶ Pop-up menu with proof rules applicable in pointer focus
- ▶ Preview of rule effect as tool tip
- ▶ Quantifier instantiation and equality rules by drag-and-drop
- ▶ Possible to hide (and unhide) parts of a sequent
- ▶ Saving and loading of proofs

Literature for this Lecture

KeYbook W. Ahrendt, B. Beckert, R. Bubel, R. Hähnle, P. Schmitt, M. Ulbrich, editors.

Deductive Software Verification - The KeY Book

Vol 10001 of *LNCS*, Springer, 2016

(E-book at link.springer.com)

- ▶ W. Ahrendt, S. Grebing, **Using the KeY Prover**
Chapter 15 in [KeYbook]

further reading:

- ▶ P.H. Schmitt, **First-Order Logic**
Chapter 2 in [KeYbook]