SKEMA
BUSINESS SCHOOL

Introduction to
Artificial Intelligence

Dmitry A. Zaitsev
http://daze.ho.ua

SKEMA BUSINESS SCHOOL

Introduction to Artificial Intelligence

Dmitry A. Zaitsev

http://daze.ho.ua

Lesson 5
Logic for AI. Propositional calculus

Lesson 5
Logic for AI. Propositional calculus

Logic in AI

A tool of analysis, a basis for knowledge representation, a programming language. Kinds of logic.

Formulae syntax and semantics

Atoms, logical connectives, and formulae syntax. Interpretation of and classification formulae.

Check inference

Truth table, algebraic transformation of formulae, inference using rules. Resolution proof technique.

Automate reasoning in Z3

Basic comands: constants, functions, assertions, satisfability check, and model.

Logic in AI

Theorem-proving and model-construction techniques

- a tool of analysis
- a basis for knowledge representation
- a programming language
- application techniques:
  - Logic programming
  - Description logics
  - Theorem proving
  - Model construction
  - Cognitive robotics
  - Merging, updating, and correcting knowledge bases

Kinds of logic

Artificial Intelligence, Logic and Formalizing Common Sense

- philosophical logic
- formalization of common sense and reasoning
- **binary logic**
- fuzzy logic
- circumscription
- modal logic
- casual logic
- temporal logic

Propositional logic

How to do

- formalize propositions as atoms
- represent information as formulae using logical connectives:
    - not
    - and
    - or
- go from information we already have to new information - reasoning or inference
    - modus ponens
    - resolution

Logical representation of knowledge

Syntax and semantics

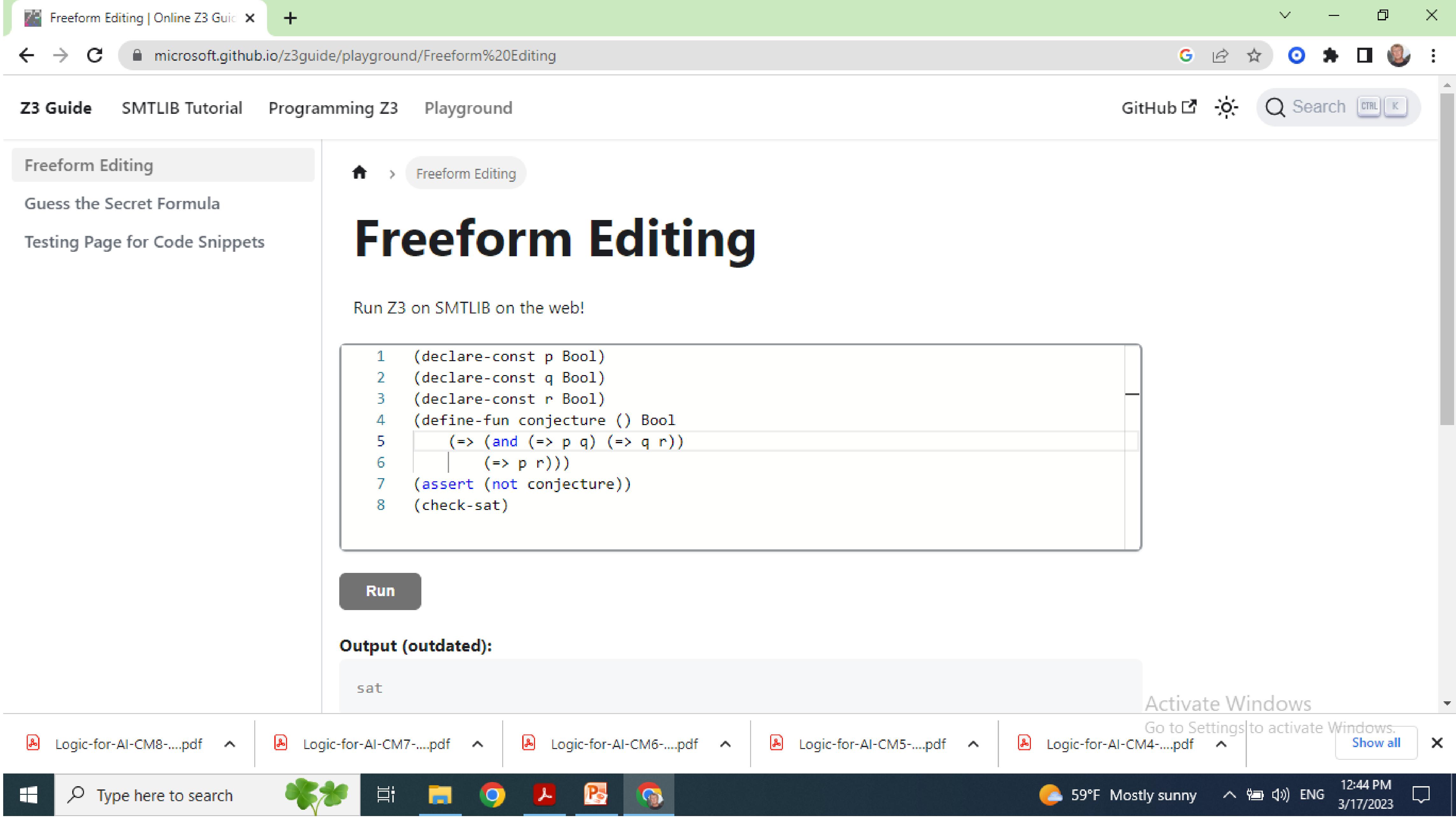
- Syntax:
  - Syntaxes are the rules which decide how we can construct legal sentences in the logic.
  - It determines which symbol we can use in knowledge representation.
  - How to write those symbols.
- Semantics:
  - Semantics are the rules by which we can interpret the sentence in the logic.
  - Semantic also involves assigning a meaning to each sentence.

Z3 theorem prover

https://microsoft.github.io/z3guide/

- Freeform Editing – online version:
  - ✓ declare constants
  - ✓ declare functions
  - ✓ make assertions
  - ✓ check satisfiability

Z3 online



Freeform Editing | Online Z3 Guide
microsoft.github.io/z3guide/playground/Freeform%20Editing
Z3 Guide
SMTLIB Tutorial
Programming Z3
Playground
GitHub
Search
Freeform Editing
Guess the Secret Formula
Testing Page for Code Snippets
Freeform Editing
Freeform Editing
Run Z3 on SMTLIB on the web!
(declare-const p Bool)
(declare-const q Bool)
(declare-const r Bool)
(define-fun conjecture () Bool
(=> (and (=> p q) (=> q r))
(=> p r)))
(assert (not conjecture))
(check-sat)
Run
Output (outdated):
sat
Activate Windows
Go to Settings to activate Windows.
Logic-for-AI-CM8-....pdf
Logic-for-AI-CM7-....pdf
Logic-for-AI-CM6-....pdf
Logic-for-AI-CM5-....pdf
Logic-for-AI-CM4-....pdf
Show all
Type here to search
59°F Mostly sunny
ENG
12:44 PM
3/17/2023

Atoms

A proposition is a declarative sentence that is either True or False, but not both

- Propositional calculus is a branch of logic that deals with propositions, which can be ***true or false***, and relations between propositions, including the construction of arguments based on them. Compound propositions are formed by connecting propositions by ***logical connectives***. Propositions that contain no logical connectives are called ***atomic propositions***. Examples:

  The sun rises in the East

  Paris is the capital of France

Logical connectives (operations)

- negation (¬P) – logical “not P”: when P is True, ¬P is False; and when P is False, ¬P is True
- conjunction (P∧Q) – logical “P and Q”: is true in only case when both are True, and is False otherwise. Other symbol &
- disjunction (P∨Q) – logical “P or Q”: is False in only case when both are False, and is True otherwise
- implication (P→Q) – material conditional “if P then Q”: Q is true whenever P is true
- identity (P↔Q) – biconditional joins "P if and only if Q“: P and Q have the same truth-value

Truth table

All combinations of arguments' values

P	$\neg P$
0	1
1	0

		and	or	exclusive or	if P then Q	P if and only if Q
P	Q	$P \wedge Q$	$P \vee Q$	$P \oplus Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	0
1	1	1	1	0	1	1

False: 0

True: 1

Conjunction check in Z3

Get a model that satisfies the function

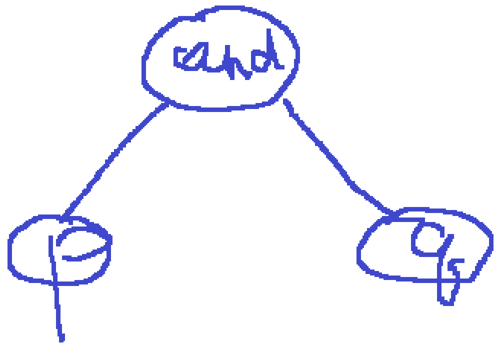
```
(declare-const p Bool)
(declare-const q Bool)
(define-fun conjecture () Bool
   (and p q))
(assert conjecture)
(check-sat)
(get-model)
```

```
sat
(
 (define-fun p () Bool
   true)
 (define-fun q () Bool
   true)
 (define-fun conjecture () Bool
   (and p q))
)
```

Using Z3 to check satisfability and tautology

Polish prefix notation of formulae

- write formula in conventional way
- create tree of formulae
- write Polish prefix notation: (operation operand1 operand2)
- when traversing tree, write node mark, then left subtree, then right subtree



Z3 basic commands

- Declare constant

  (declare-const p Bool)

- Declare function

  (define-fun conjecture () Bool (and p q))

- Assert formula – add into Z3 stack

  (assert conjecture)

- Check satisfability of all formula in stack – there is an interpretation that makes all asserted formulas true

  (check-sat)

- Get model – interpretation that makes formula satisfiable (for satisfiable formula)

  (get-model)

Valid formulae

Syntax

- atom is a valid formulae
- for any valid formulae x and y, the following formulae are valid:

  $\neg x$ and $\neg y$

  $x \land y$

  $x \lor y$

  $x \rightarrow y$

  $x \leftrightarrow y$

  (x) and (y)

Language formalization example

Assign variable symbols to propositions

“If today is Tuesday, I have a test in English or Science. If my English Professor is absent, then I will not have a test in English. Today is Tuesday and my English Professor is absent. Therefore I have a test in Science.“

T: Today is Tuesday

E: I have a test in English

S: I have a test in Science

A: My English Professor is absent

1. $T \rightarrow (E \vee S)$
2. $A \rightarrow \neg E.$
3. $T \wedge A$

$\therefore S$

Interpretation of formula

Assignment of truth values to atoms

- An interpretation of a theory is the relationship between a theory and some subject matter when there is a many-to-one correspondence between certain elementary statements of the theory, and certain statements related to the subject matter.
- The formal language for propositional logic consists of formulas built up from propositional symbols (atoms) and logical connectives. The standard kind of interpretation is a function that maps each propositional symbol to one of the truth values true or false.

Logical equivalence

Check using truth table

- Two formulae x and y are logically equivalent when for any interpretation they have the same truth values – columns of truth table coincide.
- Truth table for a formula (function) of n propositional variables contains 2^n values.
- Standard truth table considers False as 0 and True as 1 and uses increment in binary numbering system to come to the next interpretation starting from all zeroes.

Classification of formulae

Can be checked using truth table

- *satisfiable* – is true under at least one interpretation
- *tautology* – is true in any possible interpretation (is always true)
- *contradiction* – unsatisfiable statements (is always false)
- *logically contingent* – is neither a tautology nor a contradiction

Check reasoning in propositional logic

Can be checked but via NP-complete (exponential complexity) procedures

- Check tautology via composing truth table
- Equivalent algebraic transformation of formula
- Logical inference using rules of reasoning

From inference to formula – conjunction of premises implication conclusion

Check reasoning

Reasoning is correct when conjunction of premises implication conclusion is a tautology

1. $T \rightarrow (E \lor S)$
2. $A \rightarrow \neg E.$
3. $T \land A$

$\therefore S$

Compose a formula

$$(((T \rightarrow (E \lor S)) \land (A \rightarrow \neg \mathrm{E}) \land (\mathrm{T} \land \mathrm{A})) \rightarrow \mathrm{S}$$

Truth table

T	E	S	A		$E \vee S$	$T \to (E \vee S)$	$\neg E$	$A \to \neg E$	$T \wedge A$	$(\ldots)\wedge(\ldots)\wedge(\ldots)$	$(\ldots)\to S$
0	0	0	0		0	1	1	1	0	0	1
0	0	0	1		0	1	1	1	0	0	1
0	0	1	0		1	1	1	1	0	0	1
0	0	1	1		1	1	1	1	0	0	1
0	1	0	0		1	1	0	1	0	0	1
0	1	0	1		1	1	0	0	0	0	1
0	1	1	0		1	1	0	1	0	0	1
0	1	1	1		1	1	0	0	0	0	1
1	0	0	0		0	0	1	1	0	0	1
1	0	0	1		0	0	1	1	1	0	1
1	0	1	0		1	1	1	1	0	0	1
1	0	1	1		1	1	1	1	1	1	1
1	1	0	0		1	1	0	1	0	0	1
1	1	0	1		1	1	0	0	1	0	1
1	1	1	0		1	1	0	1	0	0	1
1	1	1	1		1	1	0	0	1	0	1

Check tautology in Z3

Check satiability of negation

1. $T \rightarrow (E \vee S)$
2. $A \rightarrow \neg E$.
3. $T \wedge A$

$\therefore S$

Output:

unsat

```
(declare-const t Bool)
(declare-const e Bool)
(declare-const s Bool)
(declare-const a Bool)
(define-fun conjecture () Bool
    (=> (and (=> t (or e s)) (and (=> a (not e)) (and t a))) s)
)
(assert (not conjecture))
(check-sat)
```

Algebraic laws of logic

Associativity of ∨: $x \vee (y \vee z) = (x \vee y) \vee z$

Associativity of ∧: $x \wedge (y \wedge z) = (x \wedge y) \wedge z$

Commutativity of ∨: $x \vee y = y \vee x$

Commutativity of ∧: $x \wedge y = y \wedge x$

Distributivity of ∧ over ∨: $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$

Identity for ∨: $x \vee 0 = x$

Identity for ∧: $x \wedge 1$[NB 1] $= x$

Annihilator for ∧: $x \wedge 0 = 0$

Annihilator for ∨: $x \vee 1 = 1$[NB 1]

Idempotence of ∨: $x \vee x = x$

Idempotence of ∧: $x \wedge x = x$

Absorption 1: $x \wedge (x \vee y) = x$

Absorption 2: $x \vee (x \wedge y) = x$

Distributivity of ∨ over ∧: $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$

Complementation 1 $x \wedge \neg x = 0$ [NB 1]

Complementation 2 $x \vee \neg x = 1$

De Morgan 1 $\neg x \wedge \neg y = \neg(x \vee y)$

De Morgan 2 $\neg x \vee \neg y = \neg(x \wedge y)$

Double negation $\neg(\neg x) = x$

Absorbtion 3: $\bar{x}y \vee x = y \vee x$

Equivalent algebraic transformation of formula

Replace implication: $x \to y = \bar{x} \vee y$

$(((T \to (E \vee S))(A \to \neg\mathrm{E})\mathrm{TA}) \to \mathrm{S}=$

$= \overline{(((\bar{T} \vee (E \vee S))(\bar{A} \vee \bar{E})\mathrm{TA})} \vee \mathrm{S} = \overline{(\bar{T} \vee E \vee S)} \vee \overline{(\bar{A} \vee \bar{E})} \vee \overline{TA} \vee \mathrm{S}=$

$= T\bar{E}\bar{S} \vee AE \vee \bar{T} \vee \bar{A} \vee \mathrm{S}=T\bar{E}\bar{S} \vee E \vee \bar{T} \vee \bar{A} \vee \mathrm{S}=$

$= T\bar{S} \vee E \vee \bar{T} \vee \bar{A} \vee \mathrm{S}=\bar{S} \vee E \vee \bar{T} \vee \bar{A} \vee \mathrm{S}=E \vee \bar{T} \vee \bar{A} \vee 1=1$

Algebraic style: conjunction omitted like multiplication,
negation over variables and expressions

Rules of reasoning – inference rules

Rule of Inference	Tautology	Name
p $p \to q$ $\therefore q$	$(p \wedge (p \to q)) \to q$	Modus Ponens
$\neg q$ $p \to q$ $\therefore \neg p$	$(\neg q \wedge (p \to q)) \to \neg p$	Modus Tollens
$p \to q$ $q \to r$ $\therefore p \to r$	$((p \to q) \wedge (q \to r)) \to (p \to r)$	Hypothetical syllogism
$\neg p$ $p \vee q$ $\therefore q$	$(\neg p \wedge (p \vee q)) \to q$	Disjunctive Syllogism
p $\therefore (p \vee q)$	$p \to (p \vee q)$	Addition
$(p \wedge q) \to r$ $\therefore p \to (q \to r)$	$((p \wedge q) \to r) \to (p \to (q \to r))$	Exportation
$p \vee q$ $\neg p \vee r$ $\therefore q \vee r$	$((p \vee q) \wedge (\neg p \vee r)) \to q \vee r$	Resolution

Proof by resolution technique

Resolution (and disjunctive syllogism) rule

- Prove via contradiction
- Add negation of conclusion to premises set
- Apply resolution rule, adding results to the set of formulae
- Obtain contradiction

Resolution proof example

$\bar{T} \vee E \vee S, \bar{A} \vee \bar{E}, T, A, \bar{S}$

$$\frac{\bar{T} \vee E \vee S, T}{E \vee S}$$

$$\frac{E \vee S, \bar{S}}{E}$$

$$\frac{\bar{A} \vee \bar{E}, E}{\bar{A}}$$

$$\frac{\bar{A}, A}{\blacksquare}$$

1. $T \rightarrow (E \vee S)$
2. $A \rightarrow \neg E$.
3. $T \wedge A$

$\therefore S$

Resolution rule:

$$\frac{x \vee y, \bar{x} \vee z}{y \vee z}$$

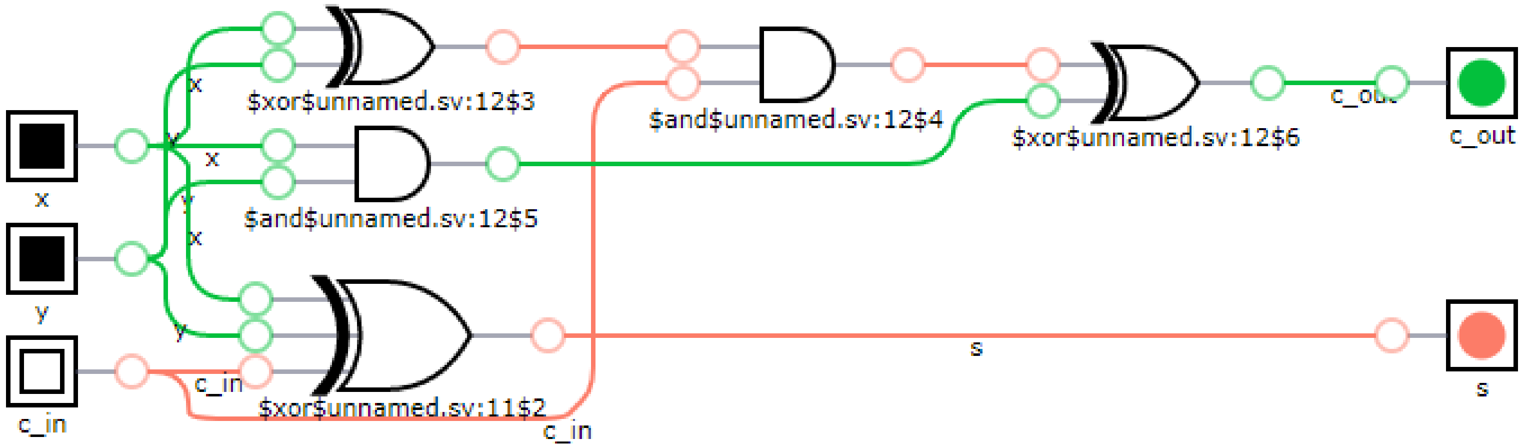
$$\frac{x, \bar{x} \vee z}{z}$$

Logic for hardware design

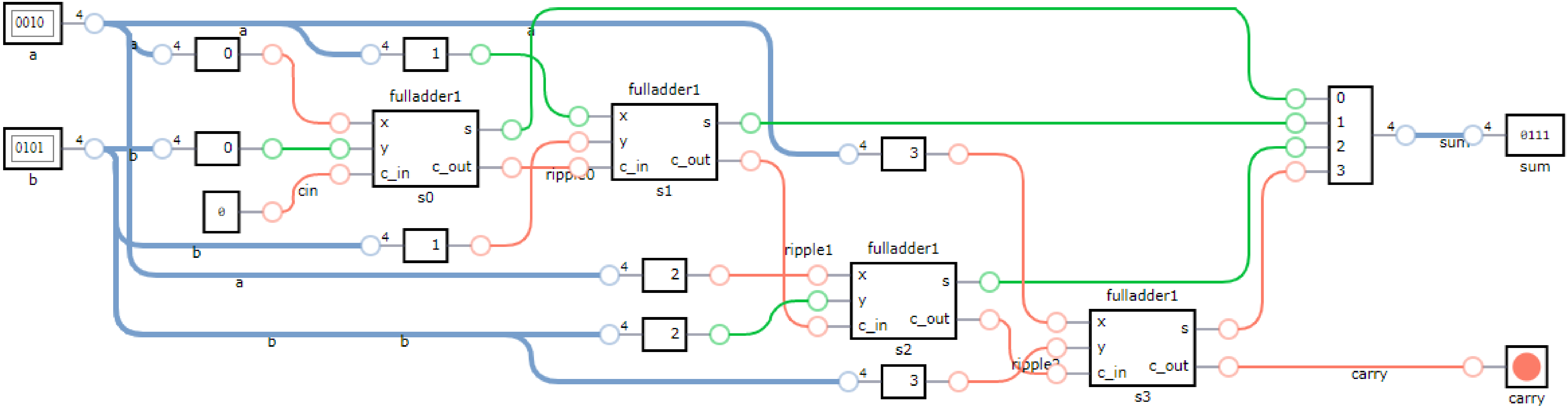
- Synthesize logic function
- Minimize logic function
- Compose scheme of gates
- An example – one bit adder:

$$s = (x \oplus y) \oplus c_{in}$$
$$c\_out = ((x \oplus y) \wedge c\_in) \oplus (x \wedge y)$$

Automated design in Verilog



Sequential adder of four bit numbers



Task 1

Propositional logic – check reasoning

- For a given reasoning offer at least two verbal interpretation in natural language
- Check reasoning manually via:
  - truth table
  - equivalent transformation of formulae
  - resolution technique
- Check reasoning in Z3

Variants for task 1

0. $P \to Q, P \vee R \vdash QR$

1. $P \to (Q \to R), \bar{R} \vdash Q$

2. $Q \vee P, P \to R, \bar{Q} \vdash R$

3. $(P \vee Q) \to R, \bar{R} \vdash \bar{Q} \vee P$

4. $PQ \vee R, Q \to R, \bar{R} \vdash P$

5. $Q \vee P, Q \to R \vdash P \to R$

6. $Q \to P, P \to R, Q \vdash R$

7. $PQ \to R, \bar{R} \vdash Q \vee P$

8. $Q \to P, Q \to R, \bar{R} \vdash P$

9. $(Q \to P) \to (Q \to R), \bar{Q} \vdash \bar{R}$

SKEMA
BUSINESS SCHOOL

Introduction to
Artificial Intelligence
Dmitry A. Zaitsev
http://daze.ho.ua

Lesson 5
Logic for AI. Propositional calculus

EFMD EMBA AACSB EQUIS skema BUSINESS SCHOOL