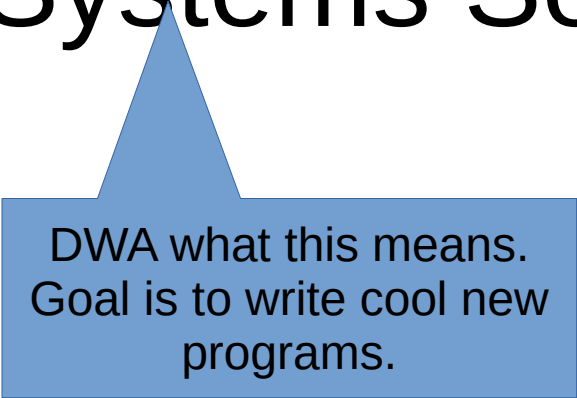


Systems Section

~~Systems~~ Section

~~Systems~~ Section



DWA what this means.
Goal is to write cool new
programs.

Hello

- Name
- Intro to neighbors

Today

- Expectations x2
- Logistics (GitHub)
- Crash course on SMT and SAT
- Overview of projects
- Intro the starter paper

Your Roles

Your Roles

- Listen & act on feedback

Your Roles

- Listen & act on feedback
- Engage with the ideas. Try to extend them.

Your Roles

- Listen & act on feedback
- Engage with the ideas. Try to extend them.
- Share your ideas

Your Roles

- Listen & act on feedback
- Engage with the ideas. Try to extend them.
- Share your ideas
- Use your resources (me !)

Your Roles

- Listen & act on feedback
- Engage with the ideas. Try to extend them.
- Share your ideas
- Use your resources (me !)
- Feel frustrated --- it's supposed to be hard !

My Role

My Role

- Give you scaffolding for **basic** project

My Role

- Give you scaffolding for **basic** project
- Point you to related work

My Role

- Give you scaffolding for **basic** project
- Point you to related work
- Explain basic ideas behind related work
(balance, but ultimately prefer this)

My Role

- Give you scaffolding for **basic** project
- Point you to related work
- Explain basic ideas behind related work
(balance, but ultimately prefer this)
- “Idea search pruning”

Logistics

- Everyone needs to be on GitHub
 - (If issue, contact me)
- Everyone needs to be added to section repo

Systems

~~Systems~~

“Automated Reasoning”

“Automated Reasoning”

- *Reasoning* = “thinking real hard”

“Automated Reasoning”

- *Reasoning* = “thinking real hard”
 - Is there any x with $x^2 = 1$ and $x < -2$?

“Automated Reasoning”

- *Reasoning* = “thinking real hard”
 - Is there any x with $x^2 = 1$ and $x < -2$?
 - Does this code have a bug?

“Automated Reasoning”

- *Reasoning* = “thinking real hard”
 - Is there any x with $x^2 = 1$ and $x < -2$?
 - Does this code have a bug?
 - Etc.

“Automated Reasoning”

- *Reasoning* = “thinking real hard”
 - Is there any x with $x^2 = 1$ and $x < -2$?
 - Does this code have a bug?
 - Etc.
- Automated reasoning = make computer do it

“Automated Reasoning”

- *Reasoning* = “thinking real hard”
 - Is there any x with $x^2 = 1$ and $x < -2$?
 - Does this code have a bug?
 - Etc.
- Automated reasoning = make computer do it
- Insight: everything is “solving equations”

“Automated Reasoning”

- *Reasoning* = “thinking real hard”
 - Is there any x with $x^2 = 1$ and $x < -2$?
 - Does this code have a bug?
 - Etc.
- Automated reasoning = make computer do it
- Insight: everything is “solving equations”
- (Broad brushes)

SMT Solvers

SMT Solvers

- Think: “equation solver.”

SMT Solvers

- Think: “equation solver.”
 - $x^2 + y^2 = 1, x + y = 1.$

SMT Solvers

- Think: “equation solver.”
 - $x^2 + y^2 = 1, x + y = 1.$
- But not just numbers

SMT Solvers

- Think: “equation solver.”
 - $x^2 + y^2 = 1, x + y = 1.$
- But not just numbers
 - $x + \text{“, World!”} = \text{“Hello, World!”}$

SMT Solvers

- Think: “equation solver.”
 - $x^2 + y^2 = 1, x + y = 1.$
- But not just numbers
 - $x + \text{“, World!”} = \text{“Hello, World!”}$
- And not just conjunctions

SMT Solvers

- Think: “equation solver.”
 - $x^2 + y^2 = 1, x + y = 1.$
- But not just numbers
 - $x + \text{“, World!”} = \text{“Hello, World!”}$
- And not just conjunctions
 - $(x + y = 1) \text{ OR } (x - y = 5)$

SMT Solvers

- Think: “equation solver.”
 - $x^2 + y^2 = 1, x + y = 1.$
- But not just numbers
 - $x + \text{“, World!”} = \text{“Hello, World!”}$
- And not just conjunctions
 - $(x + y = 1) \text{ OR } (x - y = 5)$

Try it:

<https://jfmc.github.io/z3-play/>

SAT Solvers

SAT Solvers

- Like an SMT solver, but only bool types (no numbers!)

SAT Solvers

- Like an SMT solver, but only bool types (no numbers!)
- More restricted, but can be much faster

SAT Solvers

- Like an SMT solver, but only bool types (no numbers!)
- More restricted, but can be much faster
- SAT solvers are the core of SMT solvers

SAT Solvers

- Like an SMT solver, but only bool types (no numbers!)
- More restricted, but can be much faster
- SAT solvers are the core of SMT solvers
- Uglier input; “Conjunctive Normal Form”

SAT Solvers: CNF

SAT Solvers: CNF

- SAT solver input: *conjunction of disjunctions*
 - (AND of ORs)

SAT Solvers: CNF

- SAT solver input: *conjunction of disjunctions*
 - (AND of ORs)
- Bad: $A \vee (B \wedge \neg C)$ [implicitly $== \text{true}$]

SAT Solvers: CNF

- SAT solver input: *conjunction of disjunctions*
 - (AND of ORs)
- Bad: $A \vee (B \wedge \neg C)$ [implicitly $== \text{true}$]
- Good: $(A \vee B) \wedge (A \vee \neg C)$

SAT Solvers: CNF

- SAT solver input: *conjunction of disjunctions*
 - (AND of ORs)
- Bad: $A \vee (B \wedge \neg C)$ [implicitly $== \text{true}$]
- Good: $(A \vee B) \wedge (A \vee \neg C)$
- Can always turn a formula into CNF

SAT Solvers: CNF Terminology

$$(A \vee B) \wedge (A \vee \neg C)$$

SAT Solvers: CNF Terminology

$(A \vee B) \wedge (A \vee \neg C)$

SAT Solvers: CNF Terminology

$(A \vee B) \wedge (A \vee \neg C)$

“Clauses”

SAT Solvers: CNF Terminology

$(A \vee B) \wedge (A \vee \neg C)$

SAT Solvers: CNF Terminology

$(A \vee B) \wedge (A \vee \neg C)$

SAT Solvers: CNF Terminology

$(A \vee B) \wedge (A \vee \neg C)$

“Variable,” “Constant”

SAT Solvers: CNF Terminology

$(A \vee B) \wedge (A \vee \neg C)$

SAT Solvers: CNF Terminology

$(A \vee B) \wedge (A \vee \neg C)$

SAT Solvers: CNF Terminology

$(A \vee B) \wedge (A \vee \neg C)$

“Literal”

SAT Solvers: CNF Terminology

$(A \vee B) \wedge (A \vee \neg C)$

SAT Solvers: CNF Terminology

~~$(A \vee B) \wedge (A \vee \neg C)$~~

SAT Solvers: CNF Terminology

~~$(A \vee B) \wedge (A \vee \neg C)$~~

$(A \vee B) \wedge (A \vee \neg C)$

SAT Solvers: CNF Thinking

$$(A \vee B) \wedge (A \vee \neg C)$$

SAT Solvers: CNF Thinking

$$(A \vee B) \wedge (A \vee \neg C)$$

$$\{\{1, 2\}, \{1, !3\}\}$$

SAT Solvers: CNF Thinking

$$(A \vee B) \wedge (A \vee \neg C)$$

$$\{\{1, 2\}, \{1, !3\}\}$$

Imagine numbered balls, for each number one red and one blue

SAT Solvers: CNF Thinking

$$(A \vee B) \wedge (A \vee \neg C)$$

$$\{\{1, 2\}, \{1, !3\}\}$$

Imagine numbered balls, for each number one red and one blue
Pick one ball of each number

SAT Solvers: CNF Thinking

$$(A \vee B) \wedge (A \vee \neg C)$$

$$\{\{1, 2\}, \{1, !3\}\}$$

Imagine numbered balls, for each number one red and one blue
Pick one ball of each number
Never pick red & blue of same number

SAT Solvers: CNF Thinking

$$(A \vee B) \wedge (A \vee \neg C)$$

$$\{\{1, 2\}, \{1, !3\}\}$$

Imagine numbered balls, for each number one red and one blue

Pick one ball of each number

Never pick red & blue of same number

Make sure each clause has “predicted” at least one ball you chose

SAT Solvers: CNF Thinking

$$(A \vee B) \wedge (A \vee \neg C)$$

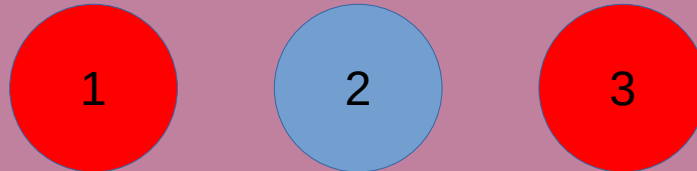
$$\{\{1, 2\}, \{1, !3\}\}$$

Imagine numbered balls, for each number one red and one blue

Pick one ball of each number

Never pick red & blue of same number

Make sure each clause has “predicted” at least one ball you chose



SAT Solvers: CNF Writing

$$(A \vee B) \wedge (A \vee \neg C)$$

$$\{\{1, 2\}, \{1, !3\}\}$$

SAT Solvers: CNF Writing

$$(A \vee B) \wedge (A \vee \neg C)$$

$\{\{1, 2\}, \{1, !3\}\}$

```
p cnf 3 2
1 2 0
1 -3 0
```

SAT Solvers: CNF Writing

$$(A \vee B) \wedge (A \vee \neg C)$$

$$\{\{1, 2\}, \{1, !3\}\}$$

```
p cnf 3 2
1 2 0
1 -3 0
```

Try it!

<https://www.comp.nus.edu.sg/~gregory/sat/>

Project 1: Proof Explorer

Project 1: Proof Explorer

- When unsat, solvers give a *proof*

Project 1: Proof Explorer

- When unsat, solvers give a *proof*
- Usually only ever read by other programs

Project 1: Proof Explorer

- When unsat, solvers give a *proof*
- Usually only ever read by other programs
- Recently, Heule et al. tried to understand one of these proofs and discovered the SMT solver found a new proof of old math theorem.

Project 1: Proof Explorer

- When unsat, solvers give a *proof*
- Usually only ever read by other programs
- Recently, Heule et al. tried to understand one of these proofs and discovered the SMT solver found a new proof of old math theorem.
- Goal: IDE/tool for humans reading these proofs

Project 1: Proof Explorer

- When unsat, solvers give a *proof*
- Usually only ever read by other programs
- Recently, Heule et al. tried to understand one of these proofs and discovered the SMT solver found a new proof of old math theorem.
- Goal: IDE/tool for humans reading these proofs

Good if: you like logic

Project 2: Auto-Incrementalizer

Project 2: Auto-Incrementalizer

- Solvers have support for solving many similar problems more quickly, “learn from past”

Project 2: Auto-Incrementalizer

- Solvers have support for solving many similar problems more quickly, “learn from past”
- But hard to use, need to make explicit exactly what is the same

Project 2: Auto-Incrementalizer

- Solvers have support for solving many similar problems more quickly, “learn from past”
- But hard to use, need to make explicit exactly what is the same
- Goal: tool to automatically identify similarities in formulas and tell the solver

Project 2: Auto-Incrementalizer

- Solvers have support for solving many similar problems more quickly, “learn from past”
- But hard to use, need to make explicit exactly what is the same
- Goal: tool to automatically identify similarities in formulas and tell the solver

Good if: you like parsing/program analysis/language design/etc.

Project 3: Evaluate Optimizations

Project 3: Evaluate Optimizations

- “Symex” tools turn program into equation

Project 3: Evaluate Optimizations

- “Symex” tools turn program into equation
- Solutions to equation \Rightarrow bug in program

Project 3: Evaluate Optimizations

- “Symex” tools turn program into equation
- Solutions to equation \Rightarrow bug in program
- Have many optimizations, not all well-justified

Project 3: Evaluate Optimizations

- “Symex” tools turn program into equation
- Solutions to equation \Rightarrow bug in program
- Have many optimizations, not all well-justified
- Goal: empirically evaluate these optimizations

Project 3: Evaluate Optimizations

- “Symex” tools turn program into equation
- Solutions to equation \Rightarrow bug in program
- Have many optimizations, not all well-justified
- Goal: empirically evaluate these optimizations

Good if: you like reading other people's code, or you like empirical evaluation work

Project 4: Break Network Protocols

Project 4: Break Network Protocols

- Sender puts message in every 2 seconds, receiver takes message out every 3 seconds. What could go wrong?

Project 4: Break Network Protocols

- Sender puts message in every 2 seconds, receiver takes message out every 3 seconds. What could go wrong?
- More complicated protocols involve notions of fairness, etc.

Project 4: Break Network Protocols

- Sender puts message in every 2 seconds, receiver takes message out every 3 seconds. What could go wrong?
- More complicated protocols involve notions of fairness, etc.
- Goal: use SMT solver to look for unfair strategies, etc. or prove none exist

Project 4: Break Network Protocols

- Sender puts message in every 2 seconds, receiver takes message out every 3 seconds. What could go wrong?
- More complicated protocols involve notions of fairness, etc.
- Goal: use SMT solver to look for unfair strategies, etc. or prove none exist

Good if: you like logic

Starter Paper

Starter Paper

- Chaff: the first 'modern' SAT solver

Starter Paper

- Chaff: the first 'modern' SAT solver
- Key: focus on raw engineering tricks

Starter Paper

- Chaff: the first 'modern' SAT solver
- Key: focus on raw engineering tricks
- Remember terminology: clause, literal, variable

Starter Paper

- Chaff: the first 'modern' SAT solver
- Key: focus on raw engineering tricks
- Remember terminology: clause, literal, variable
- Quick explainer we'll go through now: BCP

Boolean Constraint Propagation

Boolean Constraint Propagation

- Chaff always has a “partial solution” that it’s trying to work off of

Boolean Constraint Propagation

- Chaff always has a “partial solution” that it’s trying to work off of
- If clause has all but one literal F in partial sln, the remaining literal must be T.

Boolean Constraint Propagation

- Chaff always has a “partial solution” that it’s trying to work off of
- If clause has all but one literal F in partial sln, the remaining literal must be T.
- E.g., $\{\{1, 2\}, \{1, !3\}\}$ if partial solution is $1 \rightarrow F$, then we must have $2 \rightarrow T$ and $3 \rightarrow F$.

Boolean Constraint Propagation

- Chaff always has a “partial solution” that it’s trying to work off of
- If clause has all but one literal F in partial sln, the remaining literal must be T.
- E.g., $\{\{1, 2\}, \{1, !3\}\}$ if partial solution is $1 \rightarrow F$, then we must have $2 \rightarrow T$ and $3 \rightarrow F$.
- This is BCP

Starter Paper

Starter Paper

- Chaff: the first 'modern' SAT solver

Starter Paper

- Chaff: the first 'modern' SAT solver
- Key: focus on raw engineering tricks

Starter Paper

- Chaff: the first 'modern' SAT solver
- Key: focus on raw engineering tricks
- Remember terminology: clause, literal, variable

Starter Paper

- Chaff: the first 'modern' SAT solver
- Key: focus on raw engineering tricks
- Remember terminology: clause, literal, variable
- Quick explainer we'll go through now: BCP