Logic and Computability

TU Graz

SCIENCE PASSION TECHNOLOGY

Topic 1: Theories in Predicate Logic – Lazy Encoding

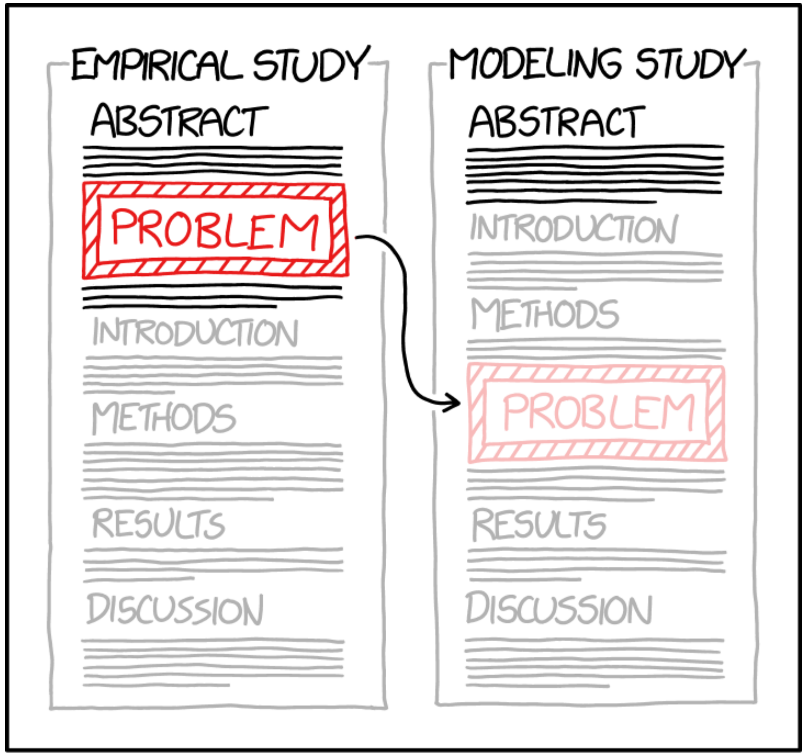
Topic 2: Symbolic Encoding

Bettina Könighofer

bettina.koenighofer@iaik.tugraz.at

Stefan Pranger

stefan.pranger@iaik.tugraz.at



A MATHEMATICAL MODEL IS A POWERFUL TOOL FOR TAKING HARD PROBLEMS AND MOVING THEM TO THE METHODS SECTION.

https://xkcd.com/2323/

Plan for Today

- **Part 1 – Lazy Encoding / DPLL(T)**

- **Part 2 – Symbolic Encoding**

Plan for Today

- **Part 1 – Lazy Encoding / DPLL(T)**
  - Recap: Theories in Predicate Logic
  - Recap: Lazy Encoding and Congruence Closure
  - Simplified Version of DPLL(T)
    - Discuss via example
- **Part 2 – Symbolic Encoding**

Plan for Today

- **Part 1 – Lazy Encoding / DPLL(T)**
  - Recap: Theories in Predicate Logic
  - Recap: Lazy Encoding and Congruence Closure
  - Simplified Version of DPLL(T)
    - Discuss via example

- **Part 2 – Symbolic Encoding**
  - Transition systems
  - Symbolic representation of sets of states
  - Symbolic representation of the transition relation
  - Symbolic encodings of arbitrary sets
  - Set operations on symbolically encoded sets

Learning Outcomes

After this lecture…

1. students can explain the simplified version of DPLL(T), especially the interaction of SAT solver and theory solver.
2. students can apply the simplified version of DPPL(T) to decide the satisfiability of formulas in $\mathcal{T}_{UFE}$.

Recap - Definition of a Theory

Definition of a First-Order Theory $\mathcal{T}$:

- Signature Σ
  - Defines the set of **constants, predicate and function symbols**
- Set of Axioms $\mathcal{A}$
  - Gives **meaning** to the predicate and function symbols

Recap - Definition of a Theory

Definition of a First-Order Theory $\mathcal{T}$:

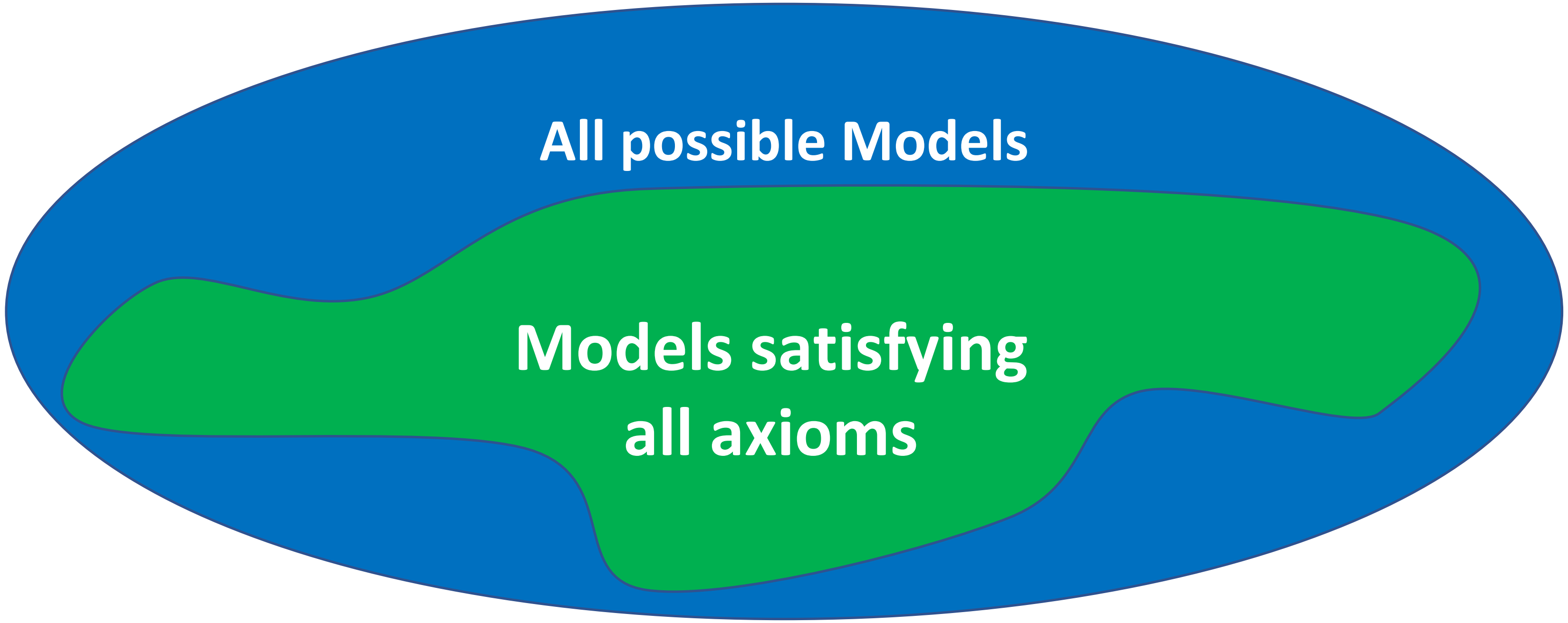
- Signature Σ
  - Defines the set of **constants, predicate and function symbols**
- Set of Axioms $\mathcal{A}$
  - Gives **meaning** to the predicate and function symbols

Example: Theory of Lineare Integer Arithmetic $\mathcal{T}_{\mathbf{LIA}}$:

- $\Sigma_{\mathrm{LIA}} := \mathbb{Z} \;\cup\; \{+,-\} \cup \{=, \neq <, \leq, >, \geq\}$
- $\mathcal{A}_{LIA}$: defines the usual meaning to all symbols
  - E.g., The function **+** is interpreted as the addition function, e.g.
    - …
    - 0+0 → 0
    - 0+1 → 1….

Recap: $\mathcal{T}$-Satisfiability, $\mathcal{T}$-validity, $\mathcal{T}$-Equivalence

- Only models satisfying axioms are relevant
- ➔ “Satisfiability ***modulo*** (=‘with respect to’) theories”



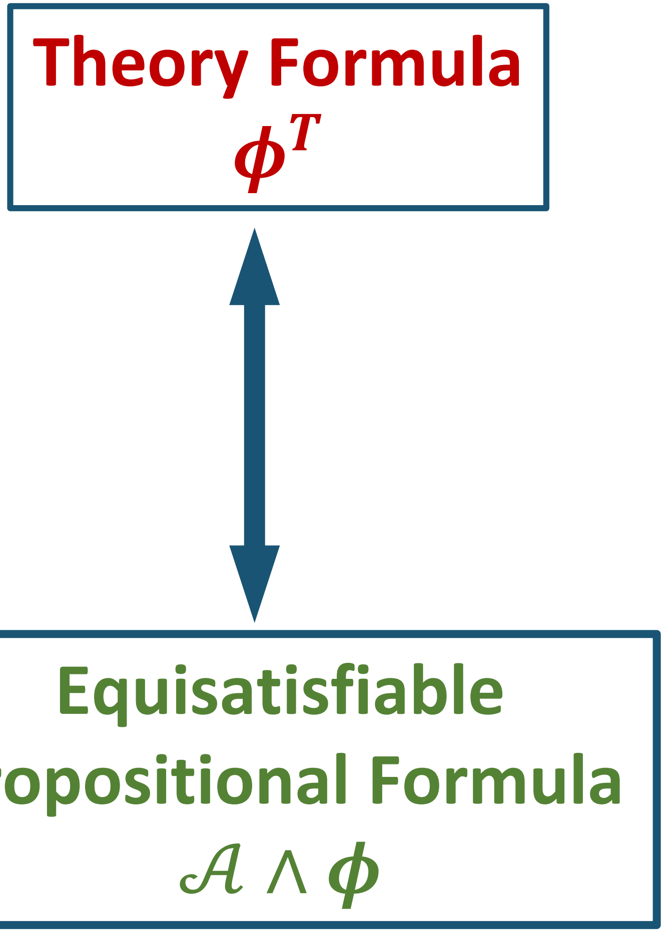
Recap - Implementations of SMT Solvers

- **Eager Encoding**

- **Lazy Encoding**

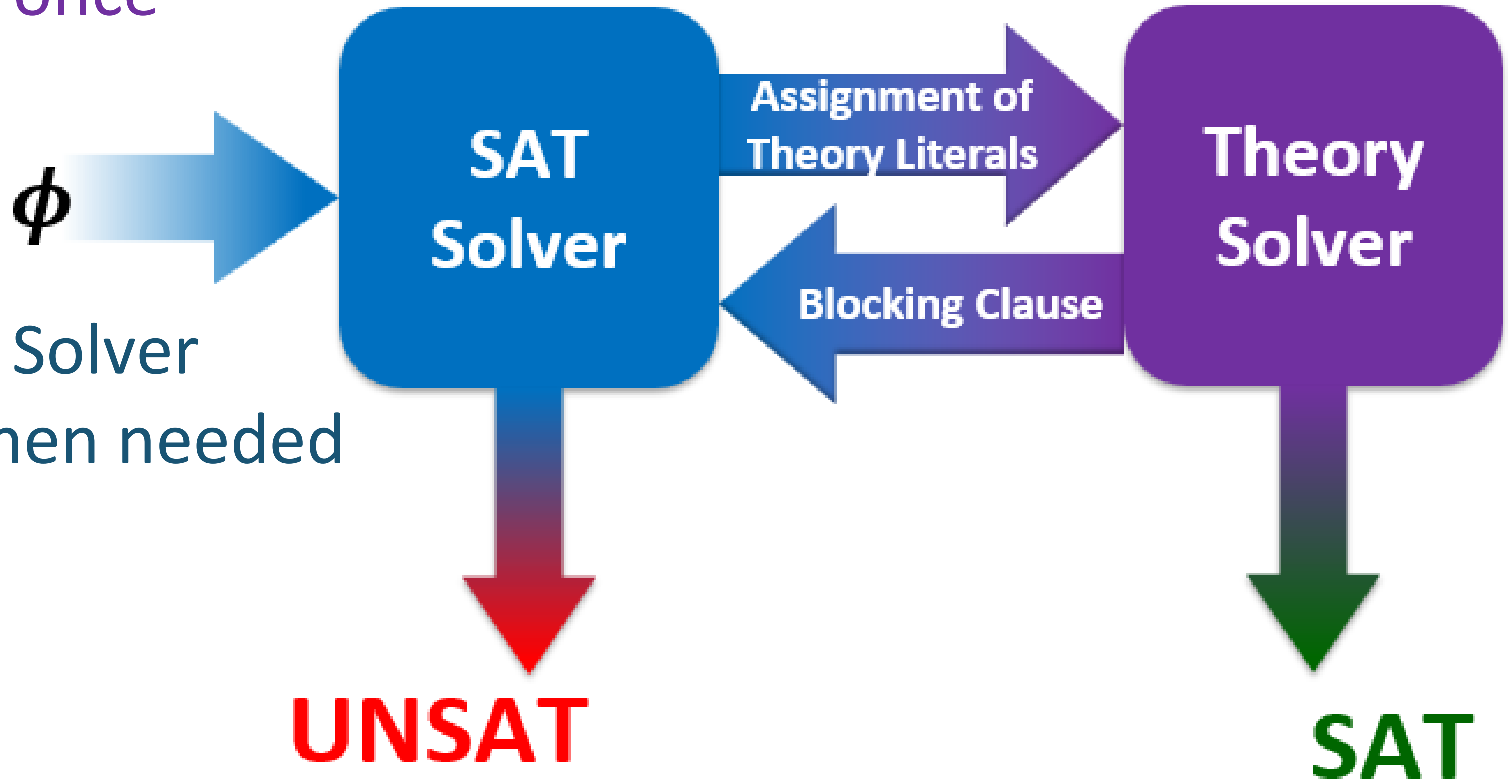
Recap - Implementations of SMT Solvers

- **Eager Encoding**
  - Equisatisfiable propositional formula
    - Adds all constraints that could be needed at once
  - SAT Solver
- **Lazy Encoding**

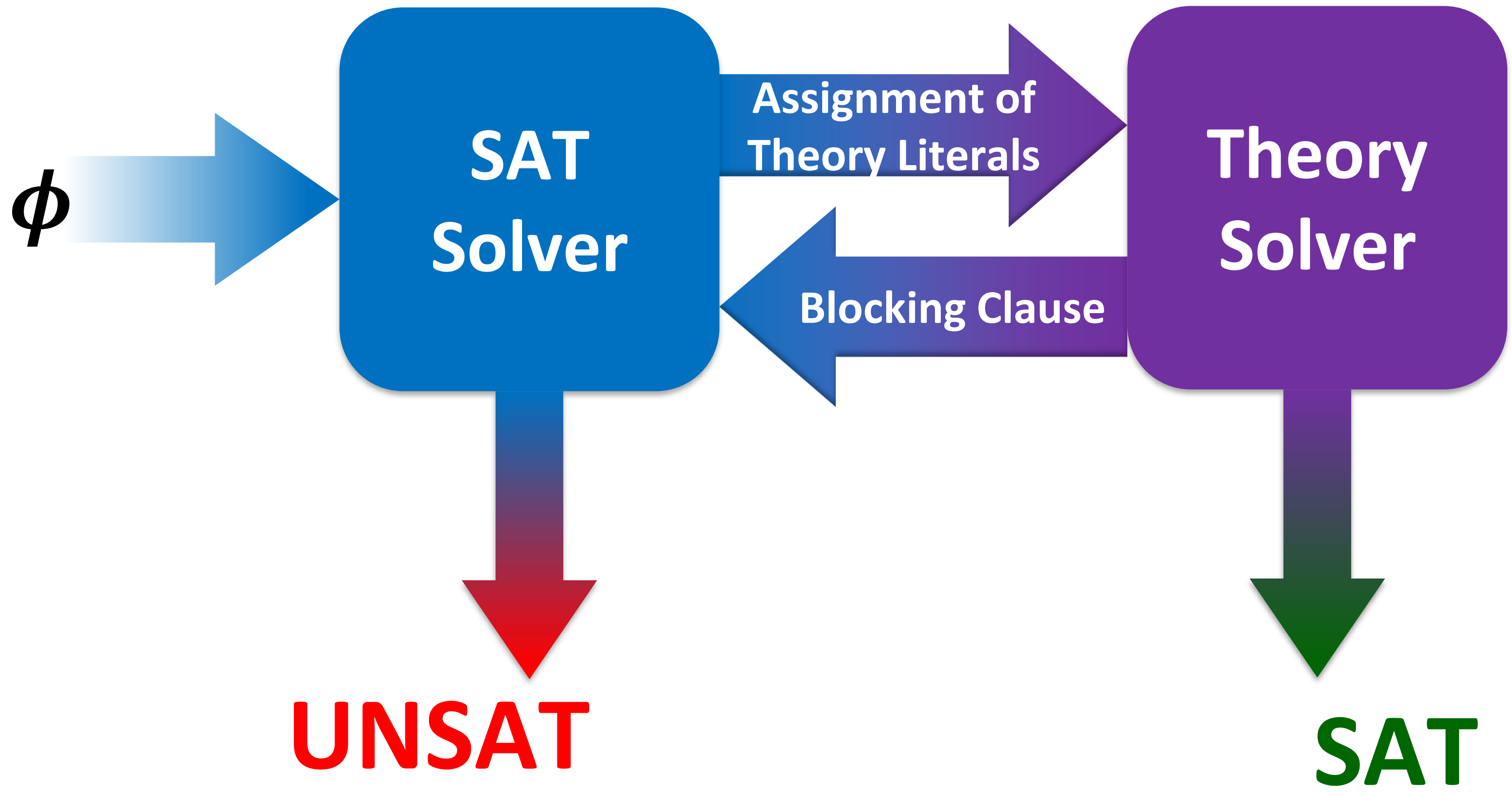


Recap - Implementations of SMT Solvers

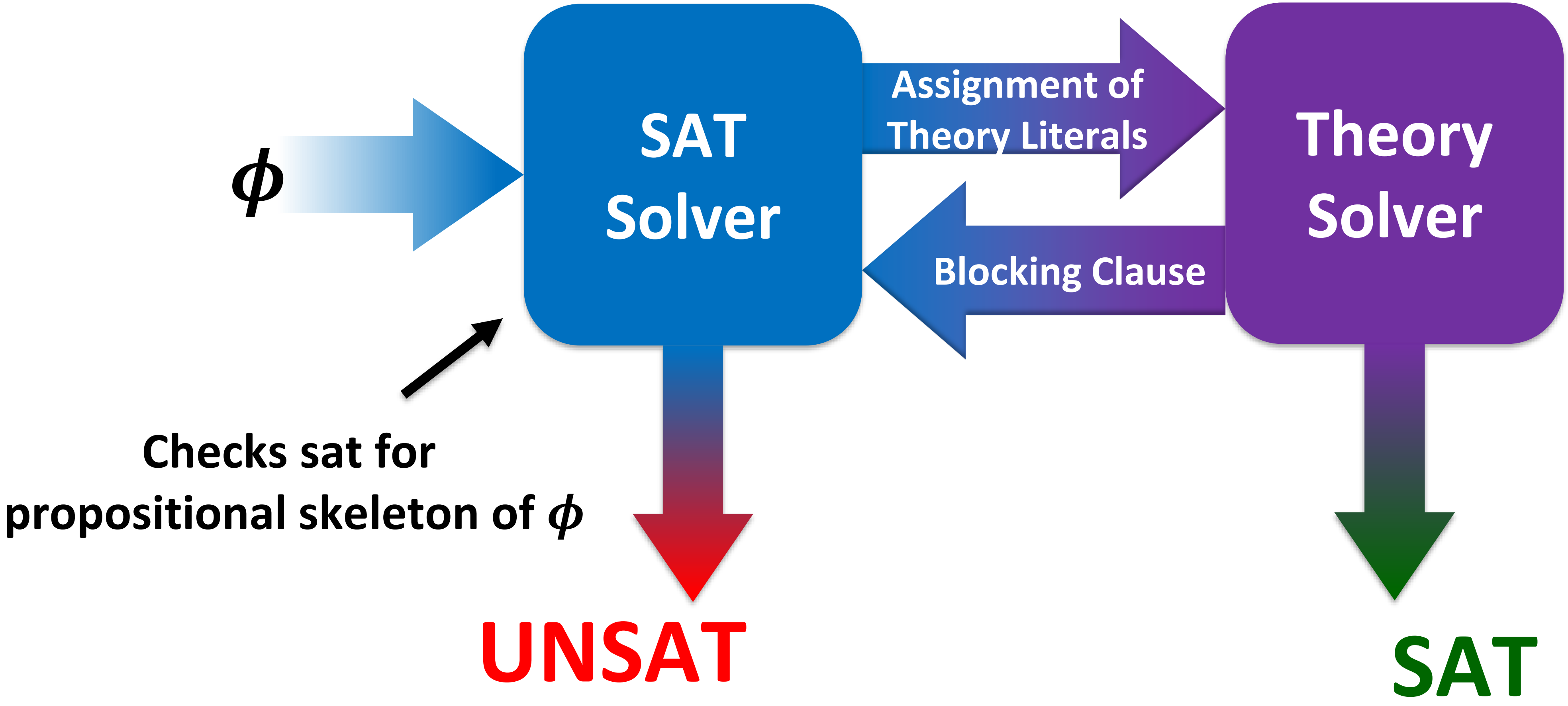
- **Eager Encoding**
  - Equisatisfiable propositional formula
    - Adds all constraints that could be needed at once
  - SAT Solver
- **Lazy Encoding**
  - SAT Solver and Theory Solver
  - Add constrains only when needed



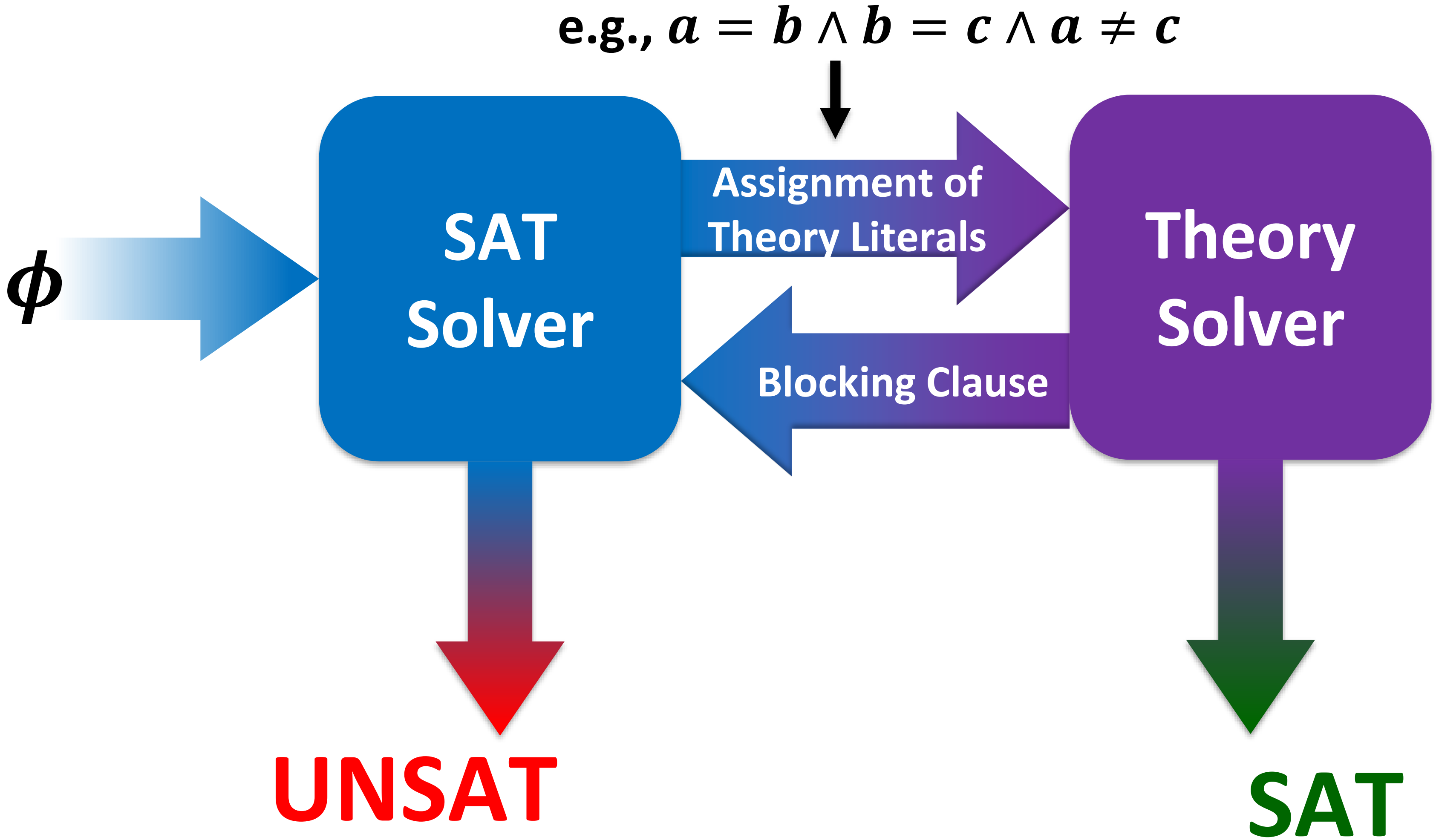
Recap - Lazy Encoding



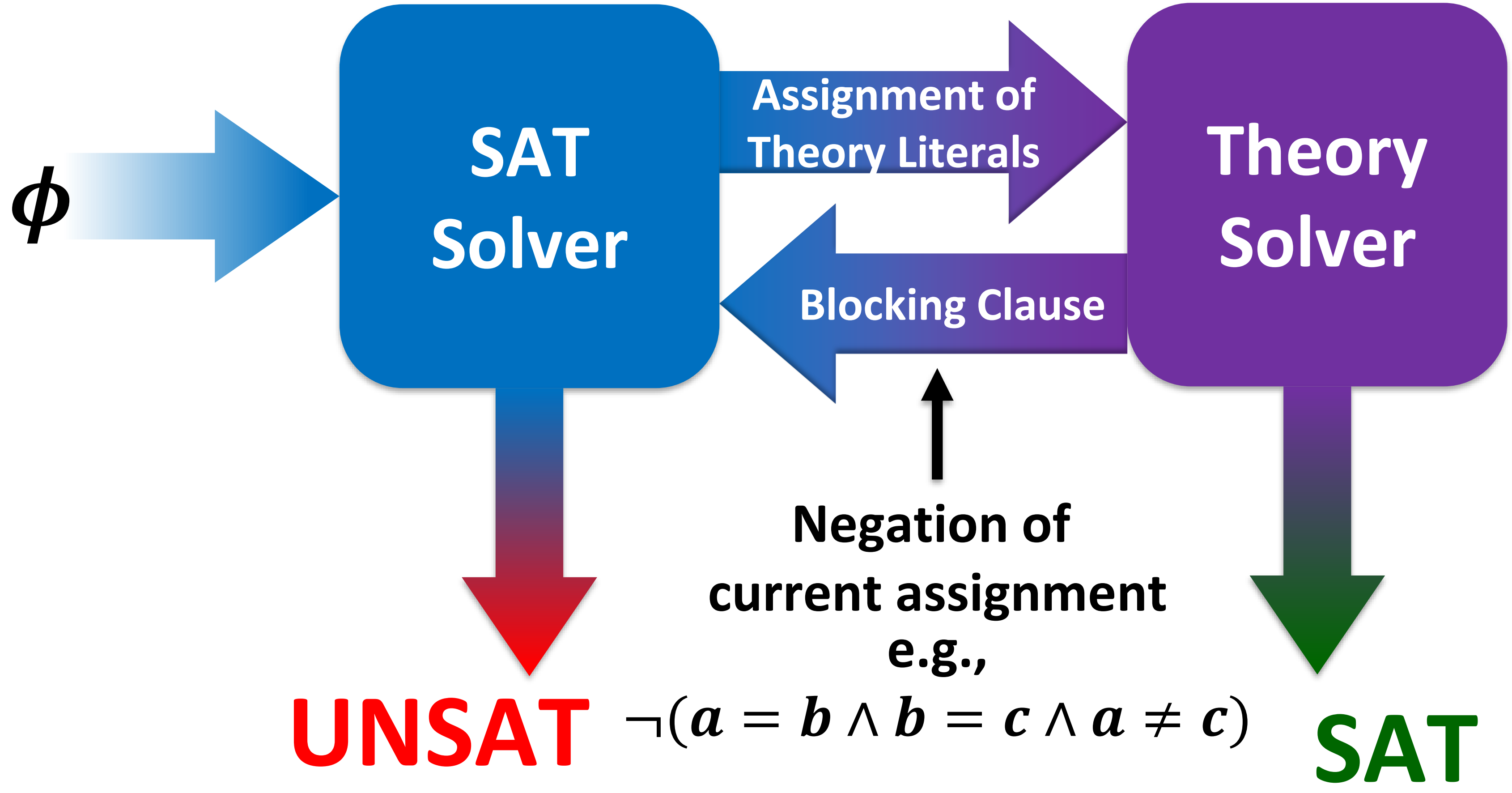
Recap - Lazy Encoding



Recap - Lazy Encoding



Recap - Lazy Encoding

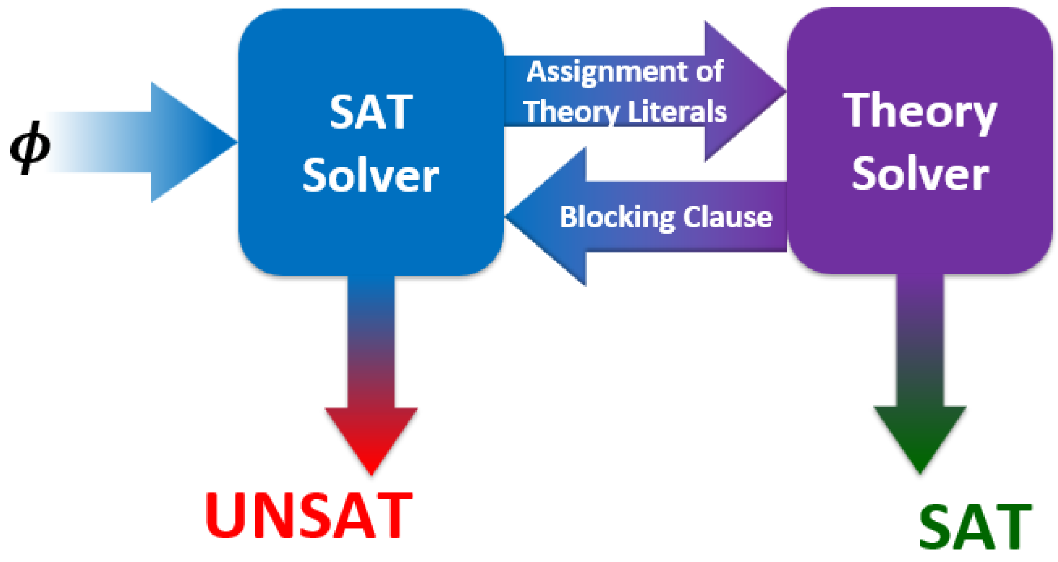


Recap – Theory Solver for $\mathcal{T}_{\mathrm{UF}E}$

Congruence Closure Algorithm

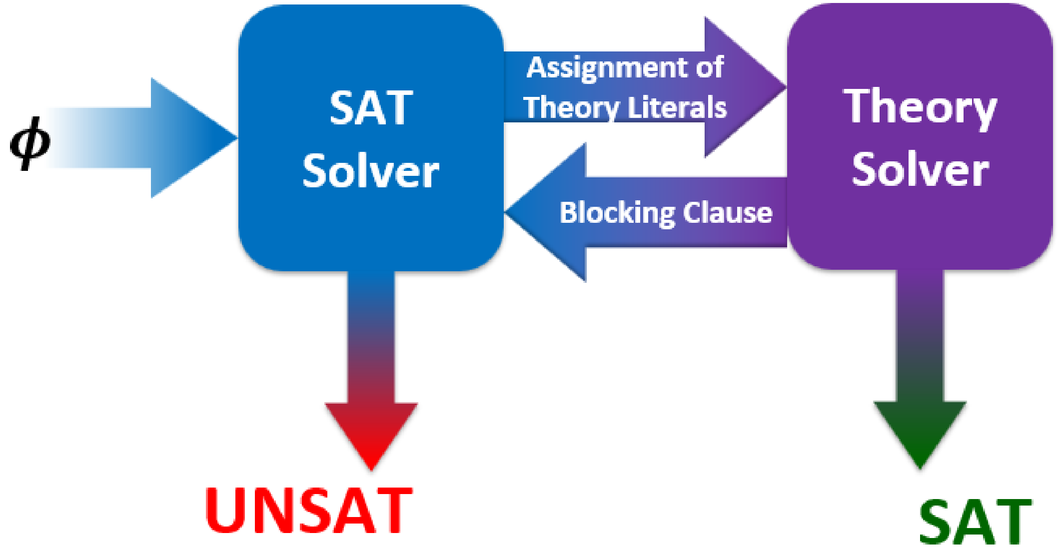
- Takes conjunctions of theory literals as input
  - Equalities (e.g., $\mathrm{f}(\mathrm{g}(\mathrm{a})) = \mathrm{g}(\mathrm{b})$)
  - Disequalities (e.g., $\mathrm{a} \neq f(b)$)

- Checks whether assignment to literals is consistent with theory
  - e.g., $a = b, b = c, c \neq a$ is $\mathcal{T}_{U\mathrm{F}E}$ unsat



Plan for Today

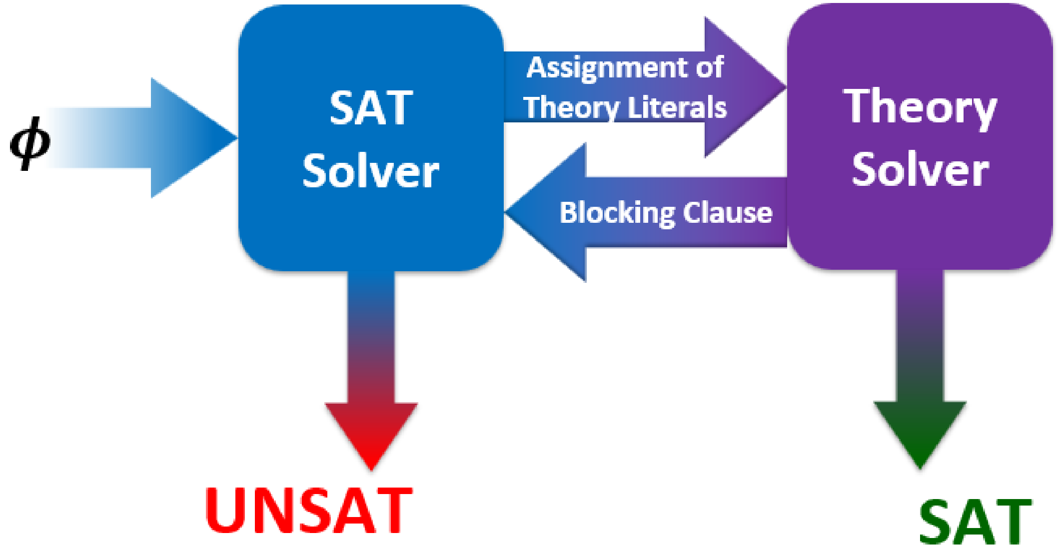
- We did not do an example for lazy encoding yet
  - → Plan for today: Examples ☺



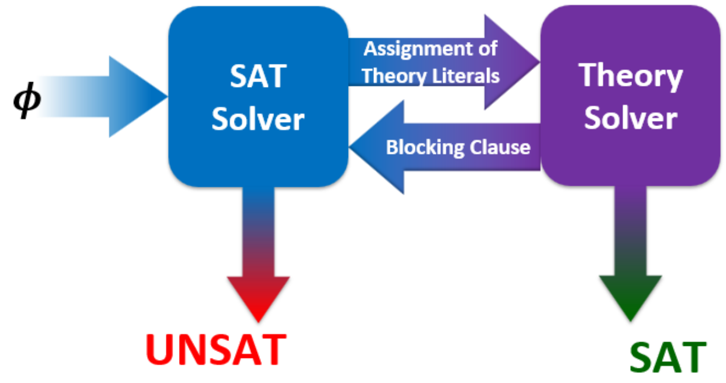
Plan for Today

- We did not do an example for lazy encoding yet
  - → Plan for today: Examples ☺

- **Deciding Satisfiability of Formulas in $\mathcal{T}_{UFE}$ using (a simplified version of) DPLL(T)**
  - Execute **DPLL with theory literals**
  - Use **Congrence Closure** to check assignment of theory literals



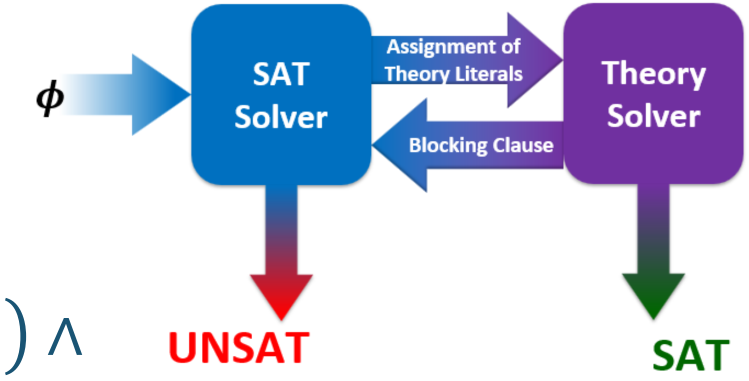
Example



Use the simple version of DPLL(T) to find satisfying assignment for φ within $\mathcal{T}_{UFE}$ (if one exists).

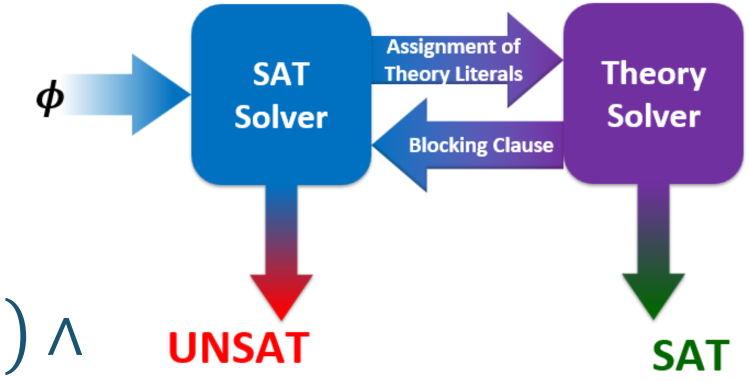
$$\varphi = ((f(g(a)) = b) \vee (f(b) = a)) \wedge ((f(g(a)) \neq b) \vee (f(b) = c)) \wedge ((f(g(a)) = b) \vee (f(a) \neq b)) \wedge ((f(b) \neq a) \vee (f(b) = c)) \wedge ((f(b) = c) \vee (f(a) = b)) \wedge ((f(b) \neq c) \vee (f(c) \neq a)) \wedge ((f(a) \neq b) \vee (f(c) \neq a))$$

Example



$$\varphi = ((f(g(a)) = b) \vee (f(b) = a)) \wedge ((f(g(a)) \neq b) \vee (f(b) = c)) \wedge$$
$$((f(g(a)) = b) \vee (f(a) \neq b)) \wedge ((f(b) \neq a) \vee (f(b) = c)) \wedge$$
$$((f(b) = c) \vee (f(a) = b)) \wedge ((f(b) \neq c) \vee (f(c) \neq a)) \wedge ((f(a) \neq b) \vee (f(c) \neq a))$$

Example

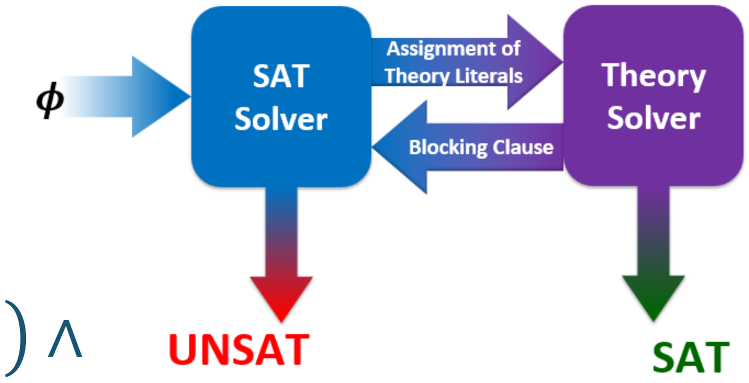


$$\varphi = ((f(g(a)) = b) \vee (f(b) = a)) \wedge ((f(g(a)) \neq b) \vee (f(b) = c)) \wedge$$
$$((f(g(a)) = b) \vee (f(a) \neq b)) \wedge ((f(b) \neq a) \vee (f(b) = c)) \wedge$$
$$((f(b) = c) \vee (f(a) = b)) \wedge ((f(b) \neq c) \vee (f(c) \neq a)) \wedge ((f(a) \neq b) \vee (f(c) \neq a))$$

- Step 1: Assign propositional variables to theory literals

$$e_0 \Leftrightarrow (f(g(a)) = b) \qquad e_3 \Leftrightarrow (f(a) = b)$$
$$e_1 \Leftrightarrow (f(b) = a) \qquad e_4 \Leftrightarrow (f(c) = a)$$
$$e_2 \Leftrightarrow (f(b) = c)$$

Example



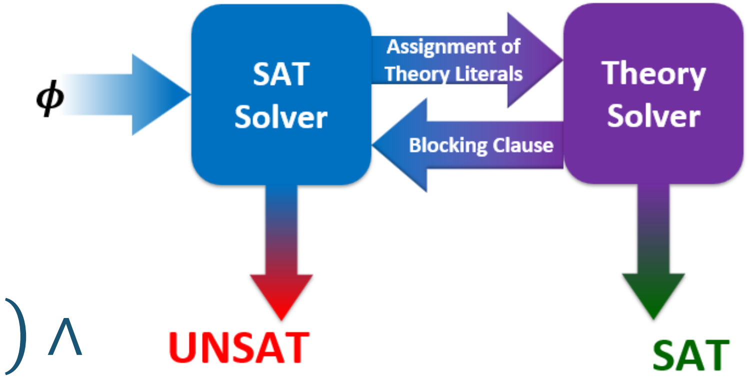
$$\varphi = ((f(g(a)) = b) \lor (f(b) = a)) \land ((f(g(a)) \neq b) \lor (f(b) = c)) \land ((f(g(a)) = b) \lor (f(a) \neq b)) \land ((f(b) \neq a) \lor (f(b) = c)) \land ((f(b) = c) \lor (f(a) = b)) \land ((f(b) \neq c) \lor (f(c) \neq a)) \land ((f(a) \neq b) \lor (f(c) \neq a))$$

- Step 1: Assign propositional variables to theory literals

$$e_0 \Leftrightarrow (f(g(a)) = b) \qquad e_3 \Leftrightarrow (f(a) = b)$$
$$e_1 \Leftrightarrow (f(b) = a) \qquad e_4 \Leftrightarrow (f(c) = a)$$
$$e_2 \Leftrightarrow (f(b) = c)$$

- Step 2: Compute propositional skeleton $\hat{\varphi}$

Example



$$\varphi = ((f(g(a)) = b) \vee (f(b) = a)) \wedge ((f(g(a)) \neq b) \vee (f(b) = c)) \wedge ((f(g(a)) = b) \vee (f(a) \neq b)) \wedge ((f(b) \neq a) \vee (f(b) = c)) \wedge ((f(b) = c) \vee (f(a) = b)) \wedge ((f(b) \neq c) \vee (f(c) \neq a)) \wedge ((f(a) \neq b) \vee (f(c) \neq a))$$

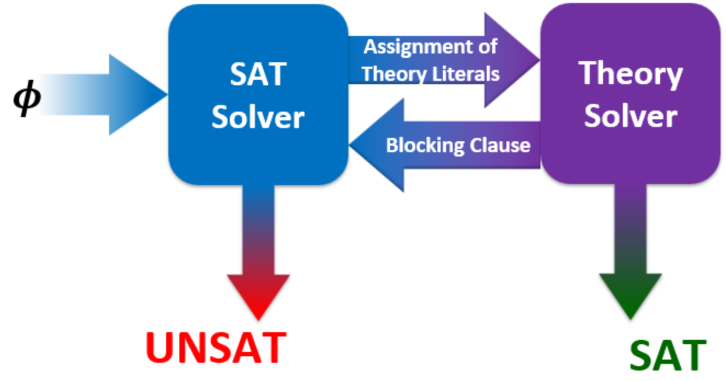
- Step 1: Assign propositional variables to theory literals

$$e_0 \Leftrightarrow (f(g(a)) = b) \quad e_3 \Leftrightarrow (f(a) = b)$$
$$e_1 \Leftrightarrow (f(b) = a) \quad e_4 \Leftrightarrow (f(c) = a)$$
$$e_2 \Leftrightarrow (f(b) = c)$$

- Step 2: Compute propositional skeleton $\hat{\varphi}$

$$\hat{\varphi} = (e_0 \vee e_1) \wedge (\neg e_0 \vee e_2) \wedge (e_0 \vee \neg e_3) \wedge (\neg e_1 \vee e_2) \wedge (e_2 \vee e_3) \wedge (\neg e_2 \vee e_4) \wedge (\neg e_3 \vee \neg e_4)$$

Example



$$\hat{\varphi} = (e_0 \vee e_1) \wedge (\neg e_0 \vee e_2) \wedge (e_0 \vee \neg e_3) \wedge (\neg e_1 \vee e_2) \wedge (e_2 \vee e_3) \wedge (\neg e_2 \vee e_4) \wedge (\neg e_3 \vee \neg e_4)$$

- Step 3: Use SAT Solver to find satisfying Model for $\hat{\varphi}$ (if one exists)

$\varphi = (e_0 \vee e_1) \wedge (\neg e_0 \vee e_2) \wedge (e_0 \vee \neg e_3) \wedge (\neg e_1 \vee e_2) \wedge (e_2 \vee e_3) \wedge (\neg e_2 \vee e_4) \wedge (\neg e_3 \vee \neg e_4)$

Decision heuristic: alphabetical order starting with the ***negative*** *phase*

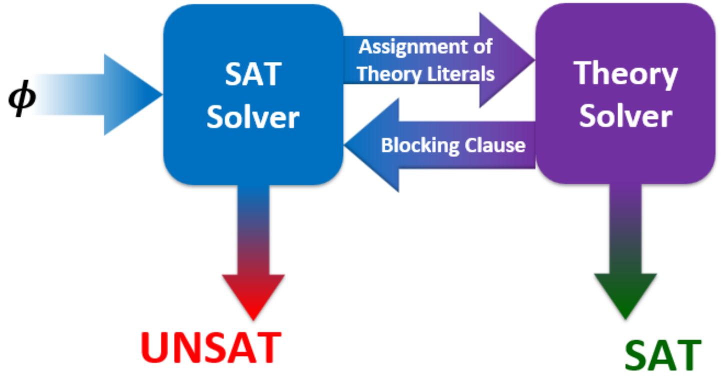
Step	1	2	3	4	5	6	7
Dec. Level							
Assignment							
1: $\{e_0, e_1\}$							
2: $\{\neg e_0, e_2\}$							
3: $\{e_0, \neg e_3\}$							
4: $\{\neg e_1, e_2\}$							
5: $\{e_2, e_3\}$							
6: $\{\neg e_2, e_4\}$							
7: $\{\neg e_3, \neg e_4\}$							
LC 1							
LC 2							
BCP							
Pure Literal							
Decision							

$\varphi = (e_0 \vee e_1) \wedge (\neg e_0 \vee e_2) \wedge (e_0 \vee \neg e_3) \wedge (\neg e_1 \vee e_2) \wedge (e_2 \vee e_3) \wedge (\neg e_2 \vee e_4) \wedge (\neg e_3 \vee \neg e_4)$

*Decision heuristic: alphabetical order starting with the **negative** phase*

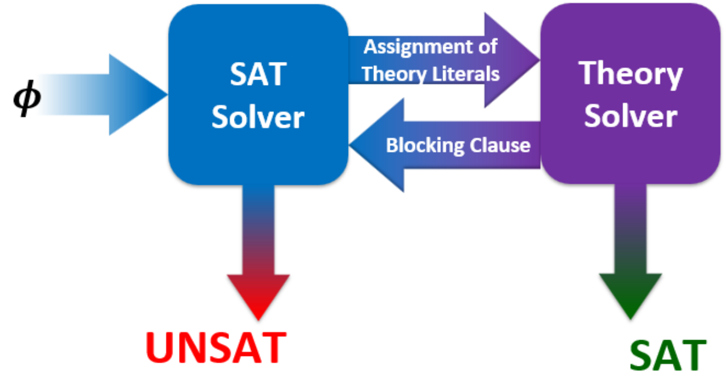
Step	1	2	3	4	5	6
Decision Level	0	1	1	1	1	1
Assignment	-	$\neg e_0$	$\neg e_0, e_1$	$\neg e_0, e_1, e_2$	$\neg e_0, e_1, e_2, \neg e_3$	$\neg e_0, e_1, e_2, \neg e_3, e_4$
Cl. 1: e_0, e_1	e_0, e_1	e_1	✓	✓	✓	✓
Cl. 2: $\neg e_0, e_2$	$\neg e_0, e_2$	✓	✓	✓	✓	✓
Cl. 3: $e_0, \neg e_3$	$e_0, \neg e_3$	$\neg e_3$	$\neg e_3$	$\neg e_3$	✓	✓
Cl. 4: $\neg e_1, e_2$	$\neg e_1, e_2$	$\neg e_1, e_2$	e_2	✓	✓	✓
Cl. 5: e_2, e_3	e_2, e_3	e_2, e_3	e_2, e_3	✓	✓	✓
Cl. 6: $\neg e_2, e_4$	$\neg e_2, e_4$	$\neg e_2, e_4$	$\neg e_2, e_4$	e_4	e_4	✓
Cl. 7: $\neg e_3, \neg e_4$	$\neg e_3, \neg e_4$	$\neg e_3, \neg e_4$	$\neg e_3, \neg e_4$	$\neg e_3, \neg e_4$	✓	✓
BCP	-	e_1	e_2	$\neg e_3$	e_4	-
PL	-	-	-	-	-	-
Decision	$\neg e_0$	-	-	-	-	SAT

Example



- Returned satisfying assignment from SAT Solver
  - $M_{prop} = \{e_0 = F, e_1 = T, e_2 = T, e_3 = F,\ e_4 = T\}$
  - $M_{prop} \vDash \hat{\varphi}$

Example

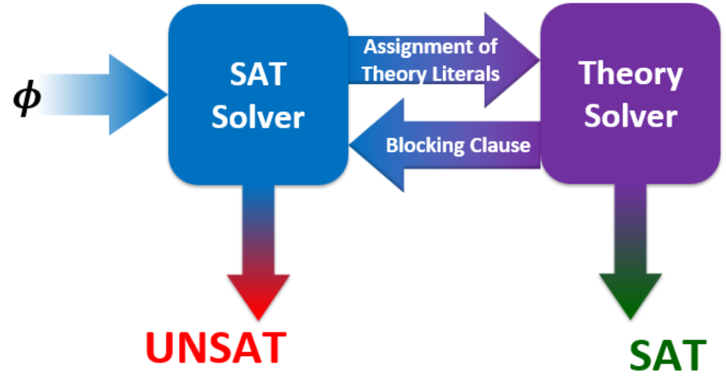


- Returned satisfying assignment from SAT Solver
  - $M_{prop} = \{e_0 = F, e_1 = T, e_2 = T, e_3 = F,\ e_4 = T\}$
  - $M_{prop} \vDash \hat{\varphi}$

- Step 4: Check if assignment of theory literals is consistent with theory
  - Translate back to theory literals using

$$e_0 \Leftrightarrow (f(g(a)) = b) \qquad e_3 \Leftrightarrow (f(a) = b)$$
$$e_1 \Leftrightarrow (f(b) = a) \qquad e_4 \Leftrightarrow (f(c) = a)$$
$$e_2 \Leftrightarrow (f(b) = c)$$

Example



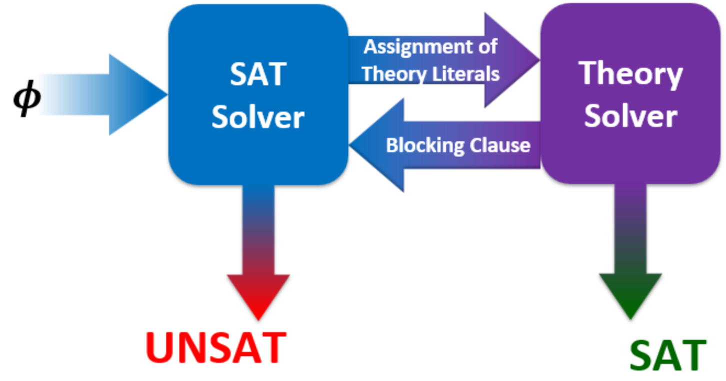
- Returned satisfying assignment from SAT Solver
  - $M_{prop} = \{e_0 = F, e_1 = T, e_2 = T, e_3 = F,\ e_4 = T\}$
  - $M_{prop} \vDash \hat{\varphi}$

- Step 4: Check if assignment of theory literals is consistent with theory
  - Translate back to theory literals using

$$e_0 \Leftrightarrow (f(g(a)) = b) \qquad e_3 \Leftrightarrow (f(a) = b)$$
$$e_1 \Leftrightarrow (f(b) = a) \qquad e_4 \Leftrightarrow (f(c) = a)$$
$$e_2 \Leftrightarrow (f(b) = c)$$

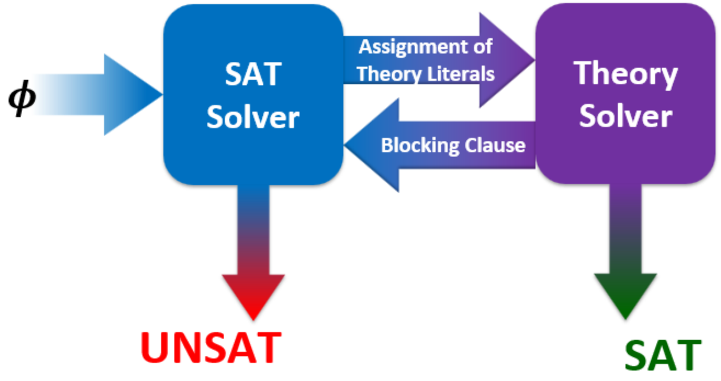
  - $M_{\mathcal{T}_{UFE}} := \{(f(g(a)) \neq\ b), (f(b) =\ a), (f(b) =\ c), (f(a) \neq\ b), (f(c) =\ a)\}$

Example



- Execute Congruence Closure Algorithm
  - $M_{\mathcal{T}_{UFE}} := \{(f(g(a)) \neq b), (f(b) = a), (f(b) = c), (f(a) \neq b), (f(c) = a)\}$

Example



- Execute Congruence Closure Algorithm
  - $M_{\mathcal{T}_{UFE}} := \{(f(g(a)) \neq b), (f(b) = a), (f(b) = c), (f(a) \neq b), (f(c) = a)\}$

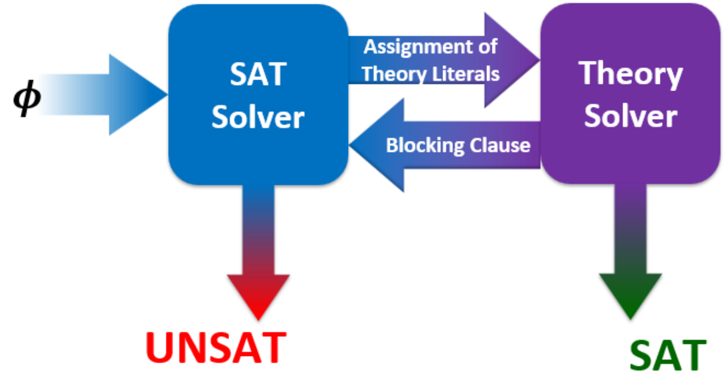
$\{f(b), a\}, \{f(b), c\}, \{f(c), a\}, \{f(g(a))\}, \{b\}, \{f(a)\}$

$\{a, c, f(b)\}, \{f(c), a\}, \{f(g(a))\}, \{b\}, \{f(a)\}$

$\{a, c, f(b), f(c)\}, \{f(g(a))\}, \{b\}, \{f(a)\}$

$\{a, c, f(a)\ f(b), f(c)\}, \{f(g(a))\}, \{b\}$

Example



- Execute Congruence Closure Algorithm
  - $M_{\mathcal{T}_{UFE}} := \{(f(g(a)) \neq b), (f(b) = a), (f(b) = c), (f(a) \neq b), (f(c) = a)\}$

$\{f(b), a\}, \{f(b), c\}, \{f(c), a\}, \{f(g(a))\}, \{b\}, \{f(a)\}$

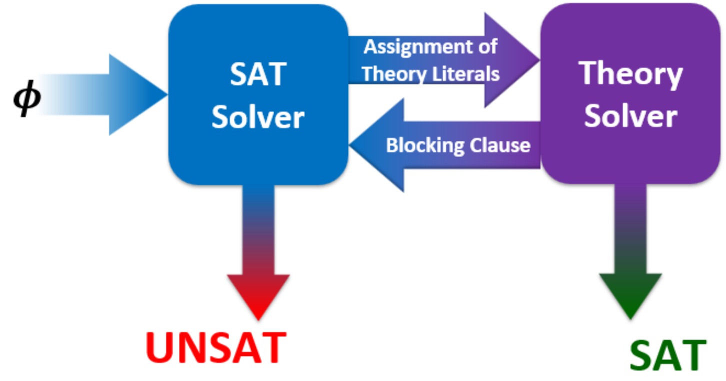
$\{a, c, f(b)\}, \{f(c), a\}, \{f(g(a))\}, \{b\}, \{f(a)\}$

$\{a, c, f(b), f(c)\}, \{f(g(a))\}, \{b\}, \{f(a)\}$

$\{a, c, f(a)\ f(b), f(c)\}, \{f(g(a))\}, \{b\}$

- $\mathcal{T}_{\mathrm{UFE}}$-Satisfiable since $f(g(a))$ and b as well as $f(a)$ and b are in different equivalence classes.

Example



- Execute Congruence Closure Algorithm
  - $M_{\mathcal{T}_{UFE}} := \{(f(g(a)) \neq b), (f(b) = a), (f(b) = c), (f(a) \neq b), (f(c) = a)\}$

$\{f(b), a\}, \{f(b), c\}, \{f(c), a\}, \{f(g(a))\}, \{b\}, \{f(a)\}$

$\{a, c, f(b)\}, \{f(c), a\}, \{f(g(a))\}, \{b\}, \{f(a)\}$

$\{a, c, f(b), f(c)\}, \{f(g(a))\}, \{b\}, \{f(a)\}$

$\{a, c, f(a)\ f(b), f(c)\}, \{f(g(a))\}, \{b\}$

- $\mathcal{T}_{\text{UFE}}$-Satisfiable since $f(g(a))$ and b as well as $f(a)$ and b are in different equivalence classes.
- **→ $M_{\mathcal{T}_{\text{UFE}}}$ is a satisfying assignment for φ. Algorithm terminates with SAT.**

Plan for Today

- Part 1 – Lazy Encoding / DPLL(T)
  - Recap: Theories in Predicate Logic
  - Recap: Lazy Encoding and Congruence Closure
  - Simplified Version of DPLL(T)
    - Discuss via example
- Part 2 – Symbolic Encoding
  - Motivation
  - Transition systems
  - Symbolic representation of sets of states
  - Symbolic representation of the transition relation
  - Symbolic encodings of arbitrary sets
  - Set operations on symbolically encoded sets

Motivation - Symbolic Encoding

- We want to reason about systems
  - → We want automatic verification of software and hardware
- Problem: Systems have huge state spaces / number of transitions

Motivation - Symbolic Encoding

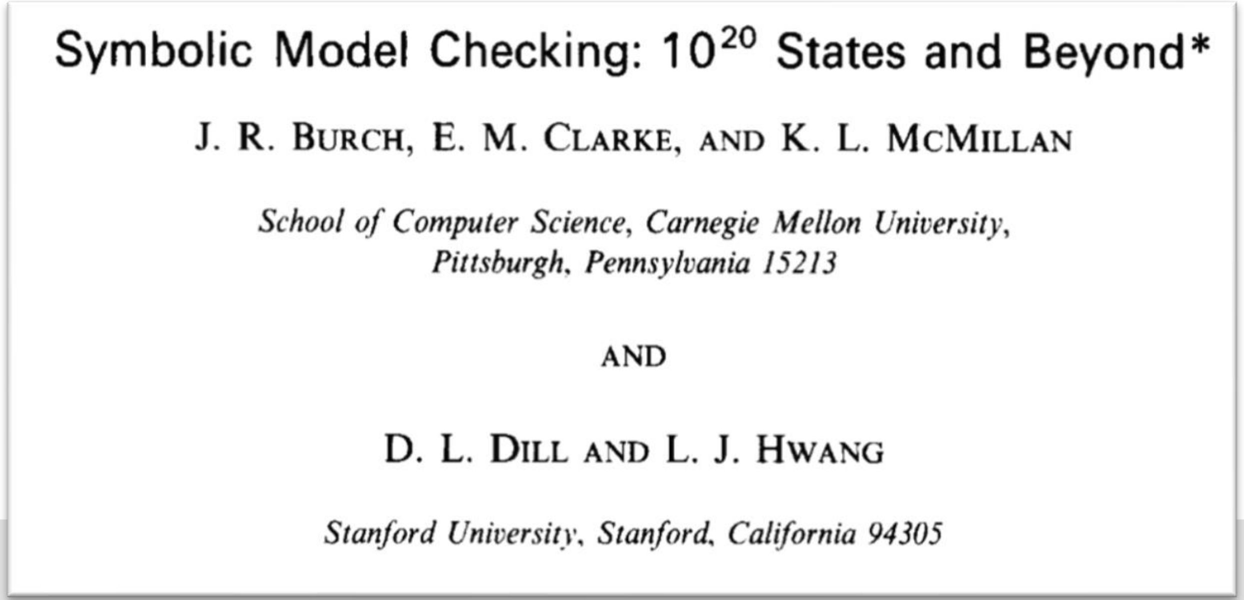
- We want to reason about systems
  - → We want automatic verification of software and hardware
- Problem: Systems have huge state spaces / number of transitions
- Automatic Verification History
  - 1981: EMC Model checker $\sim 10^4$ states

Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications

E. M. CLARKE
Carnegie Mellon University
E. A. EMERSON
University of Texas, Austin
and
A. P. SISTLA
GTE Laboratories, Inc.

Motivation - Symbolic Encoding

- We want to reason about systems
  - → We want automatic verification of software and hardware
- Problem: Systems have huge state spaces
- Automatic Verification History
  - 1981: EMC Model checker $\sim 10^4$ states
  - 1992: Symbolic Model Checking using BDDs



Symbolic Model Checking: 10^{20} States and Beyond*

J. R. BURCH, E. M. CLARKE, AND K. L. MCMILLAN

School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213

AND

D. L. DILL AND L. J. HWANG

Stanford University, Stanford, California 94305

Motivation- Symbolic Encoding

- Explicit Algorithms
  - Algorithms work **explicitly with sets** (of states and transitions)

- Symbolic Algorithms
  - Represent **sets** as **formulas**
  - Perform operations on formulas

Motivation- Symbolic Encoding

- Explicit Algorithms
  - Algorithms work **explicitly with sets** (of states and transitions)
- Symbolic Algorithms
  - Represent **sets** as **formulas**
  - Perform operations on formulas

Symbolic encoding = representation of **sets as formulas**
Symbolic set operations = logical operations on formulas representing sets

Motivation- Symbolic Encoding

- Explicit Algorithms
  - Algorithms work **explicitly with sets** (of states and transitions)

- Symbolic Algorithms
  - Represent **sets** as **formulas**
  - Perform operations on formulas
  - Advantage:
    - Often possible to represent huge sets with relatively small formulas.

Motivation- Symbolic Encoding

- Explicit Algorithms
  - Algorithms work **explicitly with sets** (of states and transitions)
- Symbolic Algorithms
  - Represent **sets** as **formulas**
  - Perform operations on formulas
- Additional Trick:
  Represent formulas via BDDs
  - Efficient representation
    & manipulation

Symbolic Model Checking: 10^{20} States and Beyond*

J. R. BURCH, E. M. CLARKE, AND K. L. MCMILLAN

School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213

AND

D. L. DILL AND L. J. HWANG

Stanford University, Stanford, California 94305

Learning Outcomes

After this lecture…

1. students can symbolically encode sets
   (in particular, sets of states and sets of transitions as well as arbitrary sets).

Learning Outcomes

After this lecture…

1. students can symbolically encode sets
   (in particular, sets of states and sets of transitions as well as arbitrary sets).
2. students can perform set operations on symbolically encoded sets.

Plan for Today

- Part 1 – Lazy Encoding / DPLL(T)
  - Recap: Theories in Predicate Logic
  - Recap: Lazy Encoding and Congruence Closure
  - Simplified Version of DPLL(T)
    - Discuss via example
- Part 2 – Symbolic Encoding
  - Motivation
  - Transition systems
  - Symbolic representation of sets of states
  - Symbolic representation of the transition relation
  - Symbolic encodings of arbitrary sets
  - Set operations on symbolically encoded sets

Transition Systems

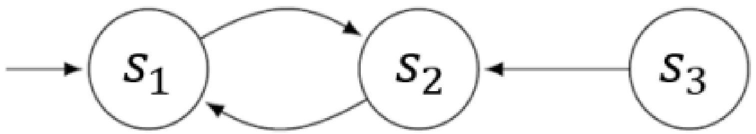
- Model of a digital system

- T is a triple (S, S_0, R)
  - Finite Set of States S
  - Set of Initial States $S_0 \subseteq S$
  - Transition Relation $R \subseteq S \times S$

Transition Systems

- Model of a digital system
- T is a triple (S, S_0, R)
  - Finite Set of States S
  - Set of Initial States $S_0 \subseteq S$
  - Transition Relation $R \subseteq S \times S$
- Often visualized as directed Graph

$S = \{s_1, s_2, s_3\}, \quad S_0 = \{s_1\}, \quad R = \{(s_1, s_2), (s_2, s_1), (s_3, s_2)\}$

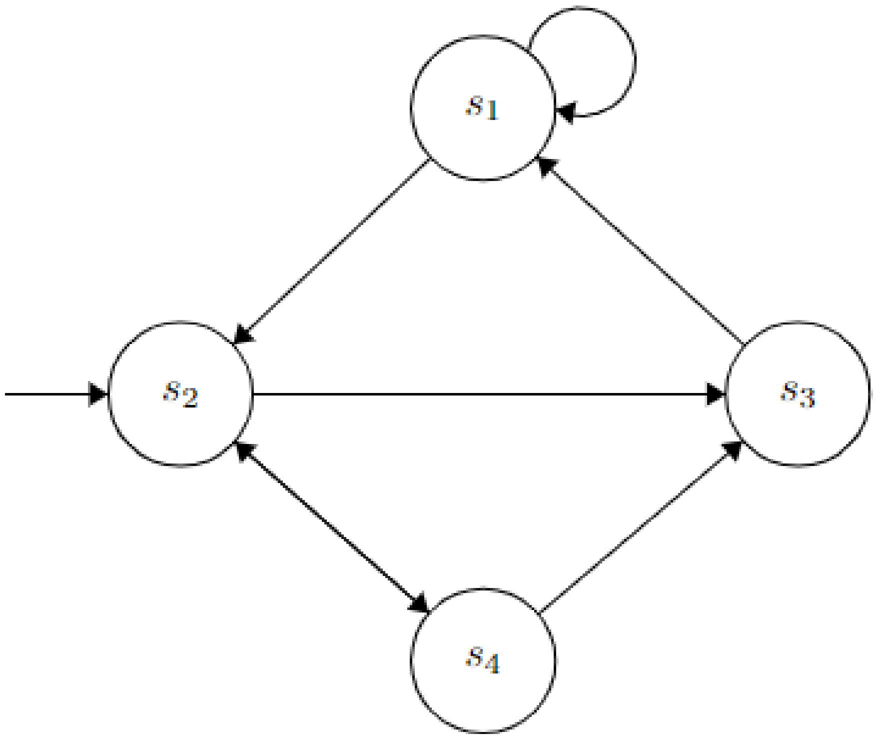


Transition Systems - Example

Draw the graph for a *transition system* $\mathcal{T}$ with: $S = \{s_1, s_2, s_3, s_4\}$,
$S_0 = \{s_2\}$,
$R = \{\{s_1, s_2\}, \{s_1, s_1\}, \{s_2, s_4\}, \{s_2, s_3\}, \{s_3, s_1\}, \{s_4, s_2\}, \{s_4, s_3\}\}$,

Transition Systems - Example

Draw the graph for a *transition system* $\mathcal{T}$ with: $S = \{s_1, s_2, s_3, s_4\}$,
$S_0 = \{s_2\}$,
$R = \{\{s_1, s_2\}, \{s_1, s_1\}, \{s_2, s_4\}, \{s_2, s_3\}, \{s_3, s_1\}, \{s_4, s_2\}, \{s_4, s_3\}\}$,



Transition Systems - Example

- Model a traffic light controller
  - Initially the red light is on. After some time, the controller switches such that the red and the yellow light are on. After some time, the controller switches to green, from green to yellow, and from yellow back to red, and so on.
- Draw the transition systems

Transition Systems - Example

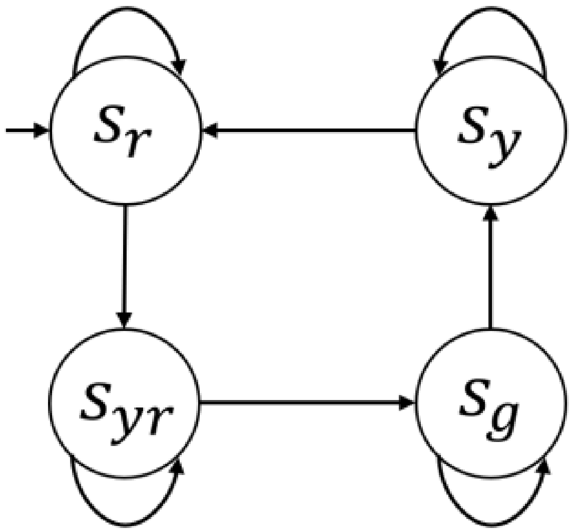
- Model a traffic light controller
  - Initially the red light is on. After some time, the controller switches such that the red and the yellow light are on. After some time, the controller switches to green, from green to yellow, and from yellow back to red, and so on.

- Draw the transition systems
  - States used:
    - s_r … the red light is on.
    - s_y … the yellow light is on.
    - s_g … the green light is on.
    - s_{ry} … the red and yellow lights are on

Transition Systems - Example

- Model a traffic light controller
  - Initially the red light is on. After some time, the controller switches such that the red and the yellow light are on. After some time, the controller switches to green, from green to yellow, and from yellow back to red, and so on.

- Draw the transition systems
  - States used:
    - s_r … the red light is on.
    - s_y … the yellow light is on.
    - s_g … the green light is on.
    - s_{ry} … the red and yellow lights are on



Plan for Today

- Part 1 – Lazy Encoding / DPLL(T)
  - Recap: Theories in Predicate Logic
  - Recap: Lazy Encoding and Congruence Closure
  - Simplified Version of DPLL(T)
    - Discuss via example
- Part 2 – Symbolic Encoding
  - Motivation
  - Transition systems
  - Symbolic representation of sets of states
  - Symbolic representation of the transition relation
  - Symbolic encodings of arbitrary sets
  - Set operations on symbolically encoded sets

Symbolic Encoding

- Systems have huge state spaces / number of transitions
- Therefore,
  - Symbolically encode sets (of states and transitions)
  - Perform set operations symbolically

Symbolic Encoding

- Systems have huge state spaces / number of transitions

- Therefore,
  - Symbolically encode sets (of states and transitions)
  - Perform set operations symbolically

- Notation
  - Use **upper-case** letters for sets
  - Use the corresponding **lower-case** letter for the formula that symbolically represents the set
    - E.g., The set F is represented via the formula f

Symbolic Representation of Sets of States

Symbolic Representation of Sets of States

- Symbolic Representation of States via Binary Encoding
  - Given $|S| \leq \mathbf{2^n}$ states, we need $\boldsymbol{n}$ Boolean variables $\{v_0, \ldots, v_{n-1}\}$ to symbolically represent the state space.

Symbolic Representation of Sets of States

- Symbolic Representation of States via Binary Encoding
  - Given $|S| \leq \mathbf{2^n}$ states, we need $\boldsymbol{n}$ Boolean variables $\{v_0, \ldots, v_{n-1}\}$ to symbolically represent the state space.

- Example: Encode the state space $S = \{s_0, s_1\}$
  - Use 1 Boolean variable v_0

Symbolic Representation of Sets of States

- Symbolic Representation of States via Binary Encoding
  - Given $|S| \leq \mathbf{2^n}$ states, we need $\boldsymbol{n}$ Boolean variables $\{v_0, \ldots, v_{n-1}\}$ to symbolically represent the state space.

- Example: Encode the state space $S = \{s_0, s_1\}$
  - Use 1 Boolean variable v_0
    - The formula $\neg \boldsymbol{v_0}$ symbolically represents the state $\boldsymbol{s_0}$
    - The formula $\boldsymbol{v_0}$ symbolically represents the state $\boldsymbol{s_1}$

Symbolic Representation of Sets of States

- Symbolic Representation of States via Binary Encoding
  - Given $|S| \leq \mathbf{2^n}$ states, we need $\boldsymbol{n}$ Boolean variables $\{v_0, \ldots, v_{n-1}\}$ to symbolically represent the state space.

- Example: Encode the state space $S = \{s_0, s_1, s_2, s_3\}$

Symbolic Representation of Sets of States

- Symbolic Representation of States via Binary Encoding
  - Given $|S| \leq \mathbf{2^n}$ states, we need $\boldsymbol{n}$ Boolean variables $\{v_0, ..., v_{n-1}\}$ to symbolically represent the state space.

- Example: Encode the state space $S = \{s_0, s_1, s_2, s_3\}$
  - Use 2 Boolean variable v_0 and v_1

Symbolic Representation of Sets of States

- Symbolic Representation of States via Binary Encoding
  - Given $|S| \leq \mathbf{2^n}$ states, we need $\boldsymbol{n}$ Boolean variables $\{v_0, \ldots, v_{n-1}\}$ to symbolically represent the state space.

- Example: Encode the state space $S = \{s_0, s_1, s_2, s_3\}$
  - Use 2 Boolean variable v_0 and v_1
    - The formula … symbolically represents the state s_0
    - The formula … symbolically represents the state s_1
    - The formula … symbolically represents the state s_2
    - The formula … symbolically represents the state s_3

Symbolic Representation of Sets of States

- Symbolic Representation of States via Binary Encoding
  - Given $|S| \leq \mathbf{2^n}$ states, we need $\boldsymbol{n}$ Boolean variables $\{v_0, \ldots, v_{n-1}\}$ to symbolically represent the state space.

- Example: Encode the state space $S = \{s_0, s_1, s_2, s_3\}$
  - Use 2 Boolean variable v_0 and v_1
    - The formula $\neg v_1 \wedge \neg v_0$ symbolically represents the state s_0
    - The formula ... symbolically represents the state s_1
    - The formula ... symbolically represents the state s_2
    - The formula ... symbolically represents the state s_3

Symbolic Representation of Sets of States

- Symbolic Representation of States via Binary Encoding
  - Given $|S| \leq \mathbf{2^n}$ states, we need $\boldsymbol{n}$ Boolean variables $\{v_0, \ldots, v_{n-1}\}$ to symbolically represent the state space.

- Example: Encode the state space $S = \{s_0, s_1, s_2, s_3\}$
  - Use 2 Boolean variable v_0 and v_1
    - The formula $\neg v_1 \land \neg v_0$ symbolically represents the state s_0
    - The formula $v_1 \land \neg v_0$ symbolically represents the state s_1
    - The formula ... symbolically represents the state s_2
    - The formula ... symbolically represents the state s_3

Symbolic Representation of Sets of States

- Symbolic Representation of States via Binary Encoding
  - Given $|S| \leq \mathbf{2^n}$ states, we need $\boldsymbol{n}$ Boolean variables $\{v_0, \ldots, v_{n-1}\}$ to symbolically represent the state space.

- Example: Encode the state space $S = \{s_0, s_1, s_2, s_3\}$
  - Use 2 Boolean variable v_0 and v_1
    - The formula $\neg v_1 \wedge \neg v_0$ symbolically represents the state s_0
    - The formula $v_1 \wedge \neg v_0$ symbolically represents the state s_1
    - The formula $\neg v_1 \wedge v_0$ symbolically represents the state s_2
    - The formula … symbolically represents the state s_3

Symbolic Representation of Sets of States

- Symbolic Representation of States via Binary Encoding
  - Given $|S| \leq \mathbf{2^n}$ states, we need $\boldsymbol{n}$ Boolean variables $\{v_0, \ldots, v_{n-1}\}$ to symbolically represent the state space.

- Example: Encode the state space $S = \{s_0, s_1, s_2, s_3\}$
  - Use 2 Boolean variable v_0 and v_1
    - The formula $\neg v_1 \wedge \neg v_0$ symbolically represents the state s_0
    - The formula $v_1 \wedge \neg v_0$ symbolically represents the state s_1
    - The formula $\neg v_1 \wedge v_0$ symbolically represents the state s_2
    - The formula $v_1 \wedge v_0$ symbolically represents the state s_3

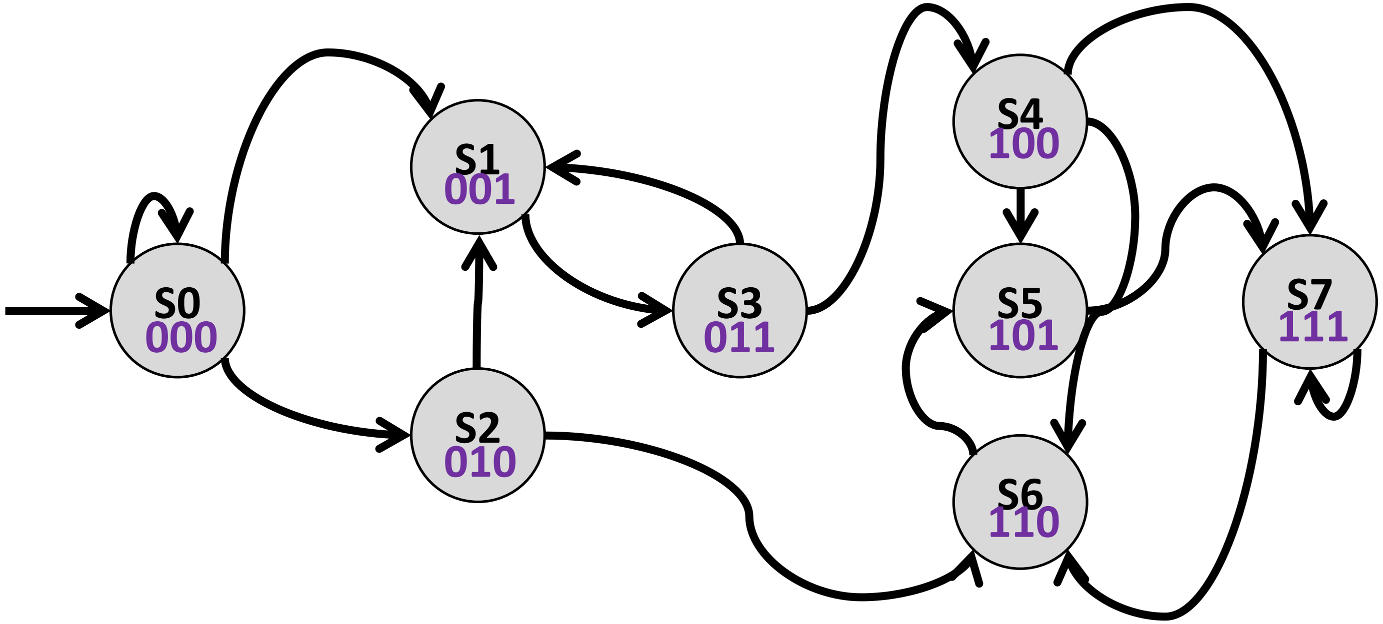
Symbolic Representation of Sets of States

- Symbolic Representation of States via Binary Encoding
  - Given $|S| \leq \mathbf{2^n}$ states, we need $\boldsymbol{n}$ Boolean variables $\{v_0, \ldots, v_{n-1}\}$ to symbolically represent the state space.

- Example: Encode the state space $S = \{s_0, s_1, s_2, s_3, s_4, \ldots, s_7\}$
  - Use 3 Boolean variable v_0 , v_1 and v_2
    - The formula $\neg v_2 \land \neg v_1 \land \neg v_0$ symbolically s_0
    - .....
    - The formula $v_2 \land v_1 \land v_0$ symbolically s_7

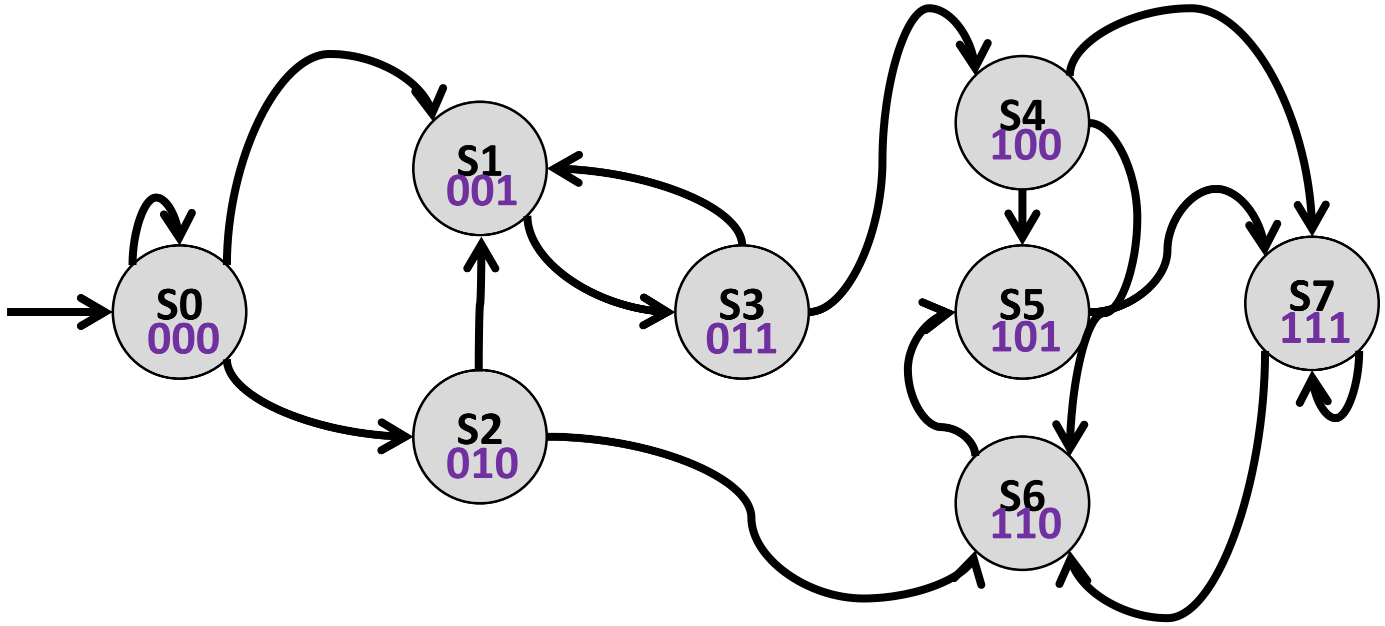
Symbolic Representation of Sets of States

- **Entire State Space:** Use variables $V = \{v_0, \ldots, v_{n-1}\}$ for binary representations of 2^nstates



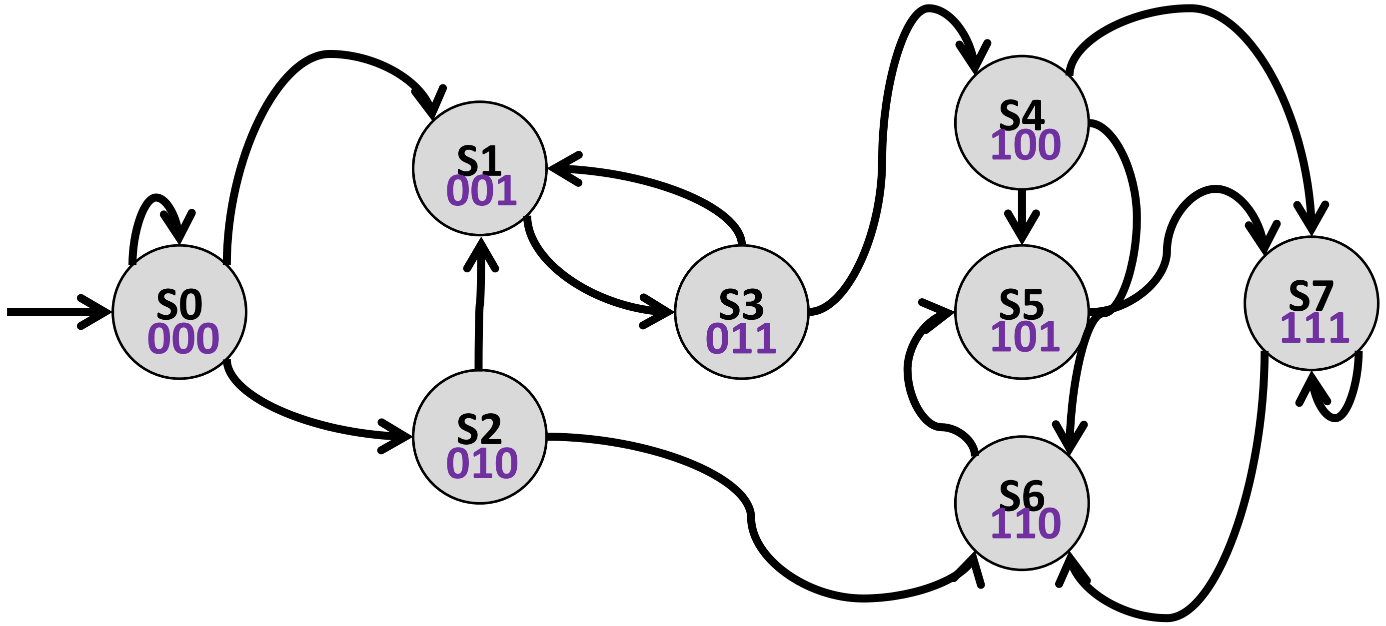
Symbolic Representation of Sets of States

- **Single State**
  - Apply binary encoding
    - E.g. State s_2 is encoded as $\neg v_2 \land v_1 \land \neg v_0$



Symbolic Representation of Sets of States

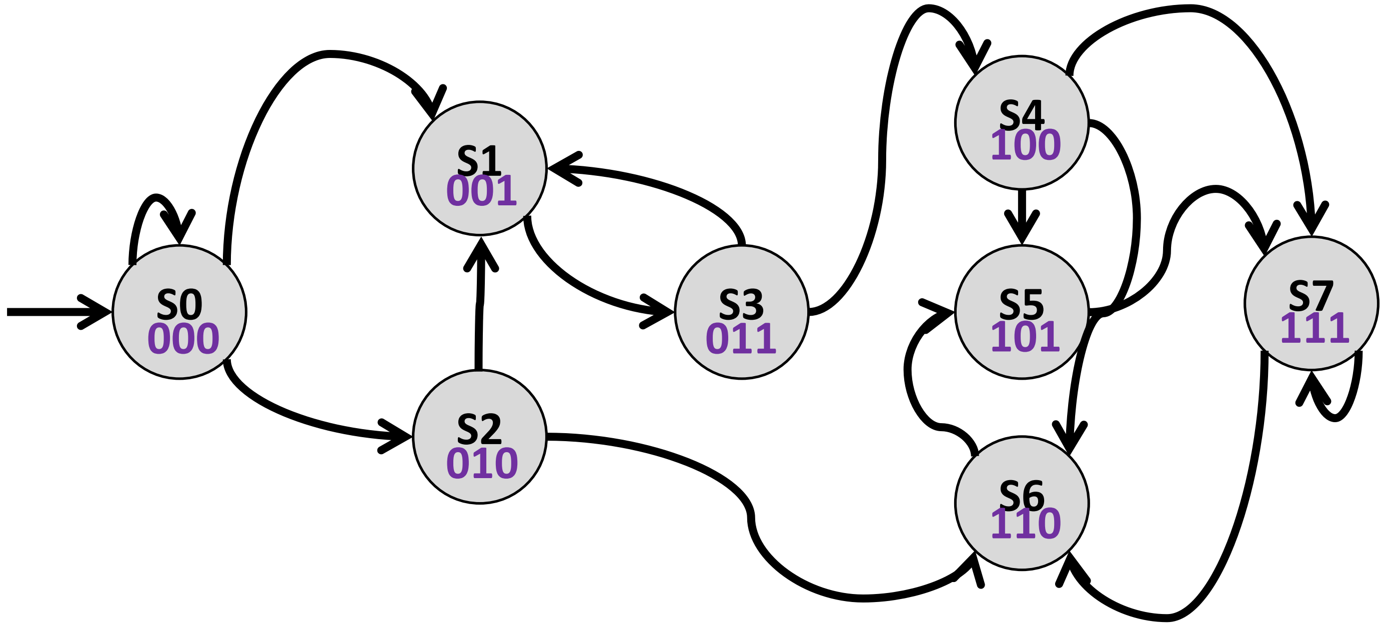
- **Sets of States**
  - Example: Symbolically encode the set of states $\{s_5, s_1\}$
    - Solution: ?



Symbolic Representation of Sets of States

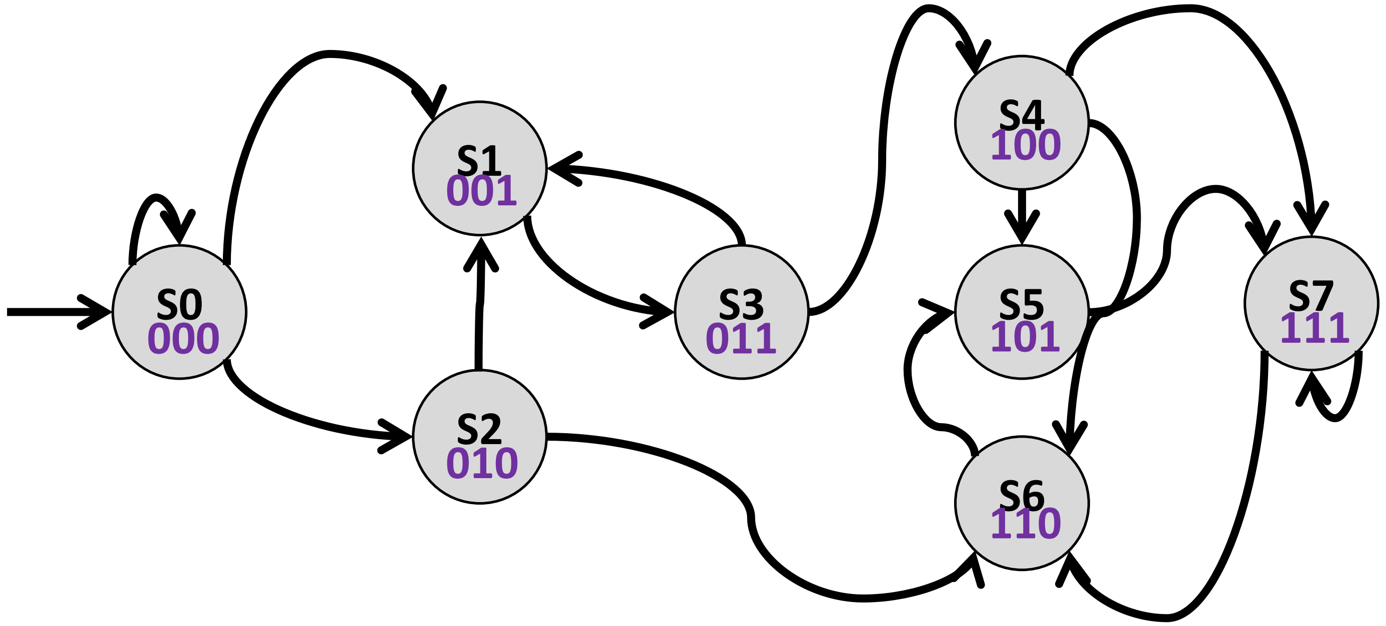
- **Sets of States**
  - Example: Symbolically encode the set of states $\{s_5, s_1\}$
    - Solution:

$$(v_2 \land \neg v_1 \land v_0) \lor (\neg v_2 \land \neg v_1 \land v_0) = \neg v_1 \land v_0$$



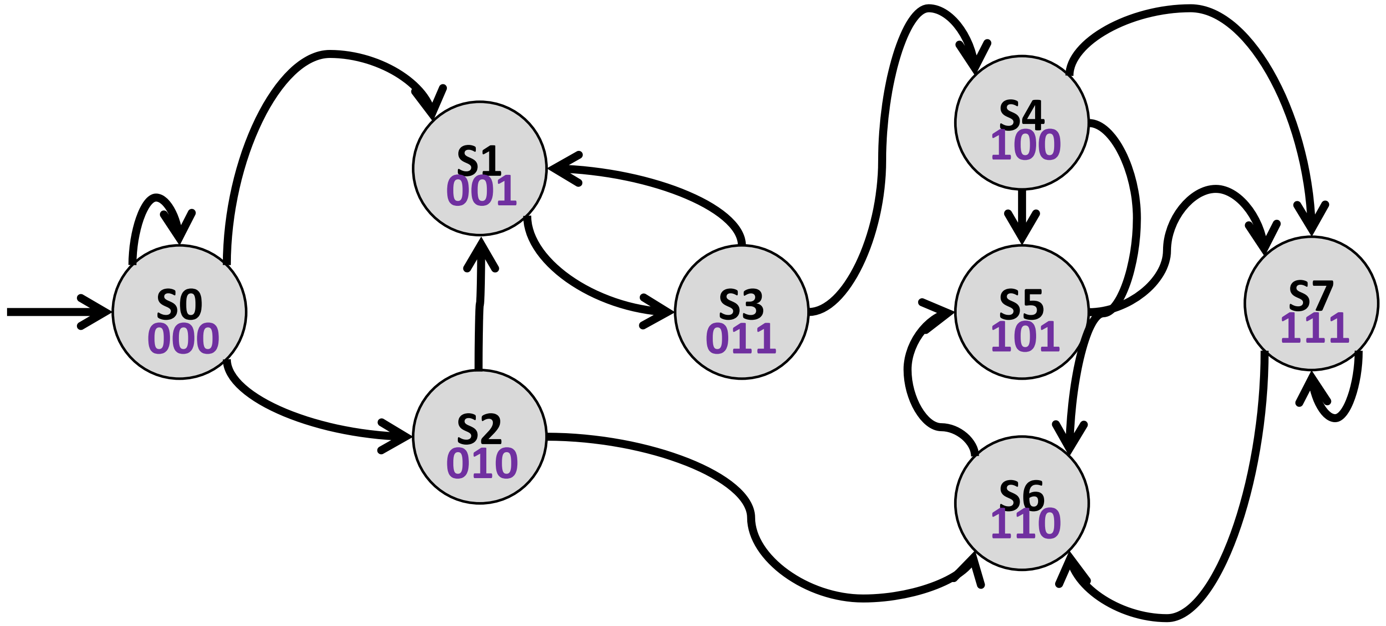
Symbolic Representation of Sets of States

- **Sets of States**
  - Example: Symbolically encode all **even numbered states**
    - Solution: ?



Symbolic Representation of Sets of States

- **Sets of States**
  - Example: Symbolically encode all **even numbered states**
    - Solution: $\neg v_0$
    - We encoded a relatively large set via a small formula. 🙂



Plan for Today

- Part 1 – Lazy Encoding / DPLL(T)
  - Recap: Theories in Predicate Logic
  - Recap: Lazy Encoding and Congruence Closure
  - Simplified Version of DPLL(T)
    - Discuss via example
- Part 2 – Symbolic Encoding
  - Motivation
  - Transition systems
  - Symbolic representation of sets of states
  - Symbolic representation of the **transition relation**
  - Symbolic encodings of arbitrary sets
  - Set operations on symbolically encoded sets

Symbolic Representation of a Single Transition

Symbolic Representation of a Single Transition

- Create a second set of variables V' (Duplicate variables)

Symbolic Representation of a Single Transition

- Create a second set of variables V' (Duplicate variables)
  - variables in $v_0, v_1, v_2, \ldots \in V$ represent **present state** variables
  - variables in $v_0', v_1', v_2', \ldots \in V'$ represent **next state** variables

Symbolic Representation of a Single Transition

- Create a second set of variables V' (Duplicate variables)
  - variables in $v_0, v_1, v_2, \ldots \in V$ represent **present state** variables
  - variables in $v'_0, v'_1, v'_2, \ldots \in V'$ represent **next state** variables

Symbolic Representation of a Single Transition

- Create a second set of variables V' (Duplicate variables)
  - variables in $v_0, v_1, v_2, \ldots \in V$ represent **present state** variables
  - variables in $v_0', v_1', v_2', \ldots \in V'$ represent **next state** variables

$$\neg v_2 \wedge \neg v_1 \wedge \neg v_0 \quad \wedge \quad \neg v_2' \wedge \neg v_1' \wedge v_0'$$

Symbolic Representation of Sets of Transitions

- Given Set of symbolically encoded edges $\mathrm{E} = \{e_1, e_2, e_3\}$

Symbolic Representation of Sets of Transitions

- Given Set of symbolically encoded edges $\mathrm{E} = \{e_1, e_2, e_3\}$

- Symbolic representation via Disjunction
  - $e = e_1 \vee e_2 \vee e_3$
  - Good for **sparse** sets of edges

Symbolic Representation of Sets of Transitions

- Given Set of symbolically encoded edges $\mathrm{E} = \{e_1, e_2, e_3\}$

- Symbolic representation via Disjunction
  - $e = e_1 \vee e_2 \vee e_3$
  - Good for **sparse** sets of edges

- Alternative
  - Exclude missing edges
    - $[\![T]\!] \setminus \{missing\ edges\}$ = Negation of union of all missing edges
    - Good for **dense** sets of edges

Symbolic Representation of Sets of Transitions

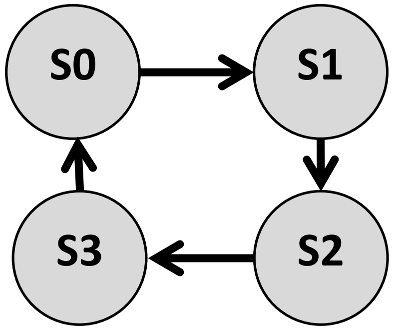
- Given Set of symbolically encoded edges $\mathrm{E} = \{e_1, e_2, e_3\}$

- Symbolic representation via Disjunction
  - $e = e_1 \vee e_2 \vee e_3$
  - Good for **sparse** sets of edges

- Alternative
  - Exclude missing edges
    - $[\![T]\!] \setminus \{missing\ edges\}$ = Negation of union of all missing edges
    - Good for **dense** sets of edges
  - Recognize patterns
    - E.g. even numbered states have edges to (all) odd numbered states
    - $\neg x_0 \wedge x_0'$

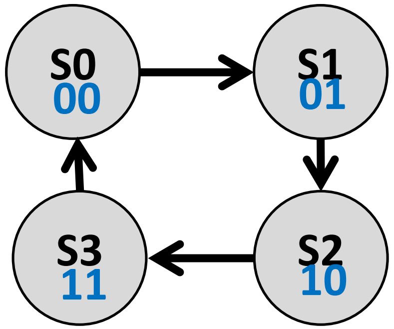
Symbolic Representation of Sets of Transitions

- Example:
  - Symbolically encode the transition relation



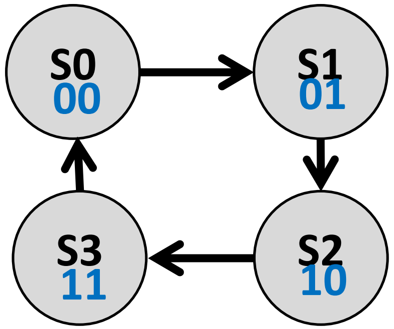
Symbolic Representation of Sets of Transitions

- Example:
  - Symbolically encode the transition relation



Symbolic Representation of Sets of Transitions

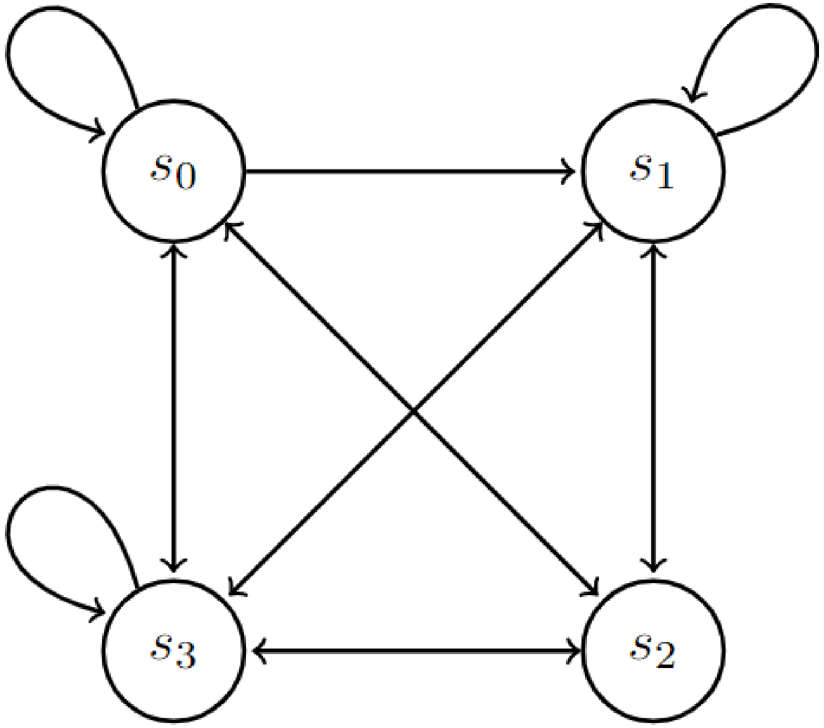
- Example:
  - Symbolically encode the transition relation



$$(\neg v_1 \land \neg v_0 \land \neg v'_1 \land v'_0) \lor$$
$$(\neg v_1 \land v_0 \land v'_1 \land \neg v'_0) \lor$$
$$(v_1 \land \neg v_0 \land v'_1 \land v'_0) \lor$$
$$(v_1 \land v_0 \land \neg v'_1 \land \neg v'_0)$$

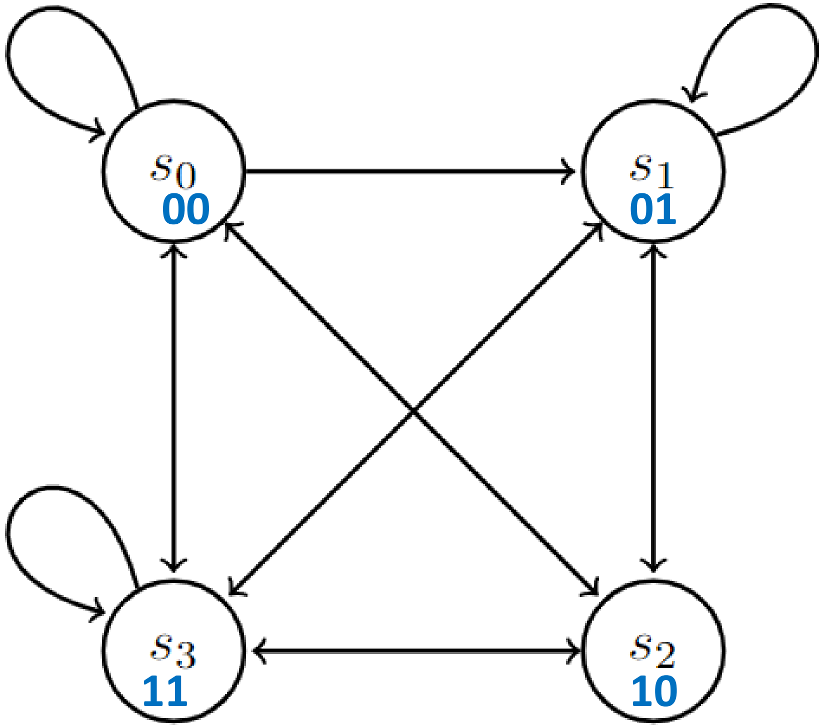
Symbolic Representation of Sets of Transitions

- Example:
  - Symbolically encode the transition relation



Symbolic Representation of Sets of Transitions

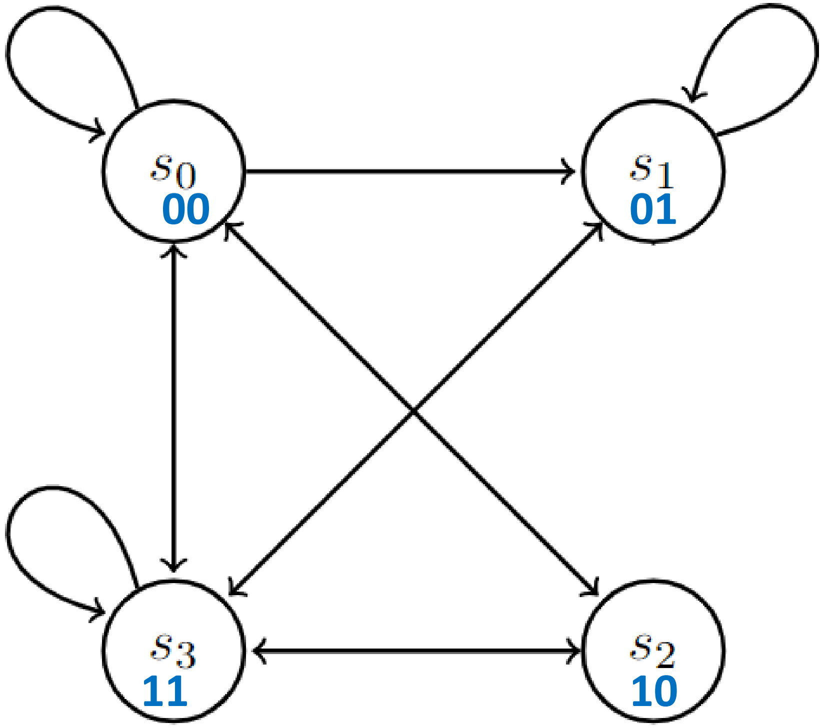
- Example:
  - Symbolically encode the transition relation



$$\neg(v_1 \land \neg v_0 \land v_1{}' \land \neg v_0{}')$$

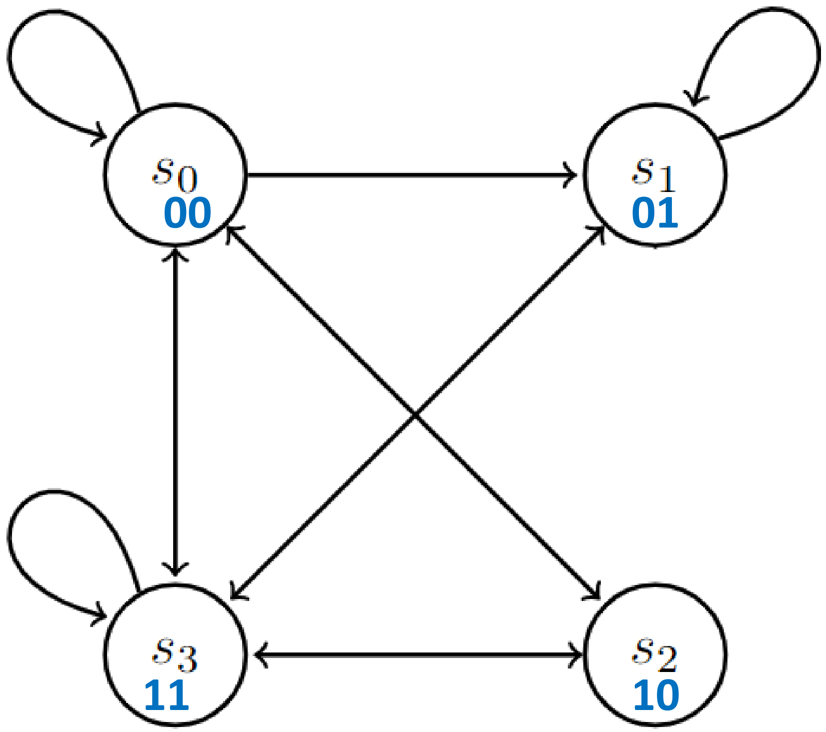
Symbolic Representation of Sets of Transitions

- Example:
  - Symbolically encode the transition relation



Symbolic Representation of Sets of Transitions

- Example:
  - Symbolically encode the transition relation



$$\neg((v_1 \land \neg v_0 \land v_1{}' \land \neg v_0{}') \lor \qquad s_2 \rightarrow s_2$$

$$(v_1 \land \neg v_0 \land \neg v_1' \land v_0') \lor \qquad s_2 \rightarrow s_1$$

$$(\neg v_1 \land v_0 \land v_1{}' \land \neg v_0{}')) \qquad s_1 \rightarrow s_2$$

Plan for Today

- Part 1 – Lazy Encoding / DPLL(T)
  - Recap: Theories in Predicate Logic
  - Recap: Lazy Encoding and Congruence Closure
  - Simplified Version of DPLL(T)
    - Discuss via example
- Part 2 – Symbolic Encoding
  - Motivation
  - Transition systems
  - Symbolic representation of sets of states
  - Symbolic representation of the **transition relation**
  - Symbolic encodings of arbitrary sets
  - Set operations on symbolically encoded sets

Symbolic Encoding of arbitrary Sets

- Domain: e.g. $\mathrm{D} = \{Austria, Germany, Spain, Italy\}$
  - $\#Vars = \lceil ld(|\mathrm{D}|) \rceil$

Symbolic Encoding of arbitrary Sets

- Domain: e.g. $\mathrm{D} = \{Austria, Germany, Spain, Italy\}$
  - $\#Vars = \lceil ld(|\mathrm{D}|) \rceil$

Element	Encoding	
	x_1	x_0
Austria		
Germany		
Spain		
Italy		

Symbolic Encoding of arbitrary Sets

- Domain: e.g. $\mathrm{D} = \{Austria, Germany, Spain, Italy\}$
  - $\#Vars = \lceil ld(|\mathrm{D}|) \rceil$

Element	Encoding	
	x_1	x_0
Austria	0	0
Germany	0	1
Spain	1	0
Italy	1	1

Symbolic Encoding of arbitrary Sets

- $F = \{Austria\}$

Element	Encoding	
	x_1	x_0
Austria	0	0
Germany	0	1
Spain	1	0
Italy	1	1

Symbolic Encoding of arbitrary Sets

- $F = \{Austria\}$
- $f = \neg x_0 \land \neg x_1$

Element	Encoding	
	x_1	x_0
Austria	0	0
Germany	0	1
Spain	1	0
Italy	1	1

Symbolic Encoding of arbitrary Sets

- $F = \{Austria\}$
- $f = \neg x_0 \land \neg x_1$

- $G = \{Austria, Spain\}$

Element	Encoding	
	x_1	x_0
Austria	0	0
Germany	0	1
Spain	1	0
Italy	1	1

Symbolic Encoding of arbitrary Sets

- $F = \{Austria\}$
- $f = \neg x_0 \wedge \neg x_1$

- $G = \{Austria, Spain\}$
- $g = \neg x_0$

Element	Encoding	
	x_1	x_0
Austria	0	0
Germany	0	1
Spain	1	0
Italy	1	1

Symbolic Encoding of arbitrary Sets

Element	Encoding	
	x_1	x_0
Austria	0	0
Germany	1	0
Spain	1	1
Italy	0	1

Element	Encoding	
	x_1	x_0
Austria	0	0
Germany	1	0
Spain	0	1
Italy	1	1

- Which encoding gives the shorter formula for the set $B = \{Germany, Spain\}$?

Symbolic Encoding of arbitrary Sets

Element	Encoding x_1	Encoding x_0
Austria	0	0
Germany	1	0
Spain	1	1
Italy	0	1

Element	Encoding x_1	Encoding x_0
Austria	0	0
Germany	1	0
Spain	0	1
Italy	1	1

- Which encoding gives the shorter formula for the set $B = \{Germany, Spain\}$?
- Answer: The first encoding:

$$f_{encoding1} = x_1 \qquad f_{encoding2} = x_1 \oplus x_0$$

Encoding Natural Numbers

- Binary Representation
- Domain $\mathbb{D}$: Usually Power of 2
  - E.g.: $\mathbb{D} = \{x \in \mathbb{N} \mid x < 2^{12}\}$

$$(457)_{10} = (0001\ 1100\ 1001)_2$$

$$x_{11} \cdots\cdots\cdots\cdots\cdots \quad x_3\ x_2\ x_1\ x_0$$

Plan for Today

- Part 1 – Lazy Encoding / DPLL(T)
  - Recap: Theories in Predicate Logic
  - Recap: Lazy Encoding and Congruence Closure
  - Simplified Version of DPLL(T)
    - Discuss via example
- Part 2 – Symbolic Encoding
  - Motivation
  - Transition systems
  - Symbolic representation of sets of states
  - Symbolic representation of the **transition relation**
  - Symbolic encodings of arbitrary sets
  - Set **operations** on symbolically encoded sets

Symbolic Operations

- **Intersection:** $F \cap G \Leftrightarrow f \wedge g$
- **Union:** $F \cup G \Leftrightarrow\ ?$
- **Difference:** $F \setminus G \Leftrightarrow ?$
- **Equality:** $F = G \Leftrightarrow ?$
- **Subset:** $F \subseteq G \Leftrightarrow\ ?$

Symbolic Operations

- **Intersection:** $F \cap G \Leftrightarrow f \wedge g$
- **Union:** $F \cup G \Leftrightarrow f \vee g$
- **Difference:** $F \setminus G \Leftrightarrow ?$
- **Equality:** $F = G \Leftrightarrow ?$
- **Subset:** $F \subseteq G \Leftrightarrow ?$

Symbolic Operations

- **Intersection:** $F \cap G \Leftrightarrow f \wedge g$
- **Union:** $F \cup G \Leftrightarrow f \vee g$
- **Difference:** $F \setminus G \Leftrightarrow f \wedge \neg g$
- **Equality:** $F = G \Leftrightarrow ?$
- **Subset:** $F \subseteq G \Leftrightarrow ?$

Symbolic Operations

- **Intersection:** $F \cap G \Leftrightarrow f \wedge g$
- **Union:** $F \cup G \Leftrightarrow f \vee g$
- **Difference:** $F \setminus G \Leftrightarrow f \wedge \neg g$
- **Equality:** $F = G \Leftrightarrow f \leftrightarrow g$
- **Subset:** $F \subseteq G \Leftrightarrow\ ?$

Symbolic Operations

- **Intersection:** $F \cap G \Leftrightarrow f \wedge g$
- **Union:** $F \cup G \Leftrightarrow f \vee g$
- **Difference:** $F \setminus G \Leftrightarrow f \wedge \neg g$
- **Equality:** $F = G \Leftrightarrow f \leftrightarrow g$
- **Subset:** $F \subseteq G \Leftrightarrow f \rightarrow g$

Example

- Domain: $A = \{x \in \mathbb{N} | 0 \leq x \leq 1023\}$
  10 bit binary representation $x_9 x_8 \ldots . x_0$
- $B = \{x \in A | x < 512\}$
- $C = \{x \in A | 256 \leq x < 768\}$

- $D = B \cup C$
- $E = B \cap C$
- $F = A \setminus E$

- TODO: Compute the symbolic representations for $B, C, D, E,$ and F

Example

- Domain: $A = \{x \in \mathbb{N} | 0 \leq x \leq 1023\}$

  10 bit binary representation $x_9 x_8 \dots . x_0$

- $B = \{x \in A | x < 512\}, b = \neg x_9$
- $C = \{x \in A | 256 \leq x < 768\}, c = (\neg x_9 \wedge x_8) \vee (x_9 \wedge \neg x_8)$?

- $D = B \cup C$
- $E = B \cap C$
- $F = A | E$

256 010.... 0
511 011.... 1
512 100.... 0
767 101.... 1

Example

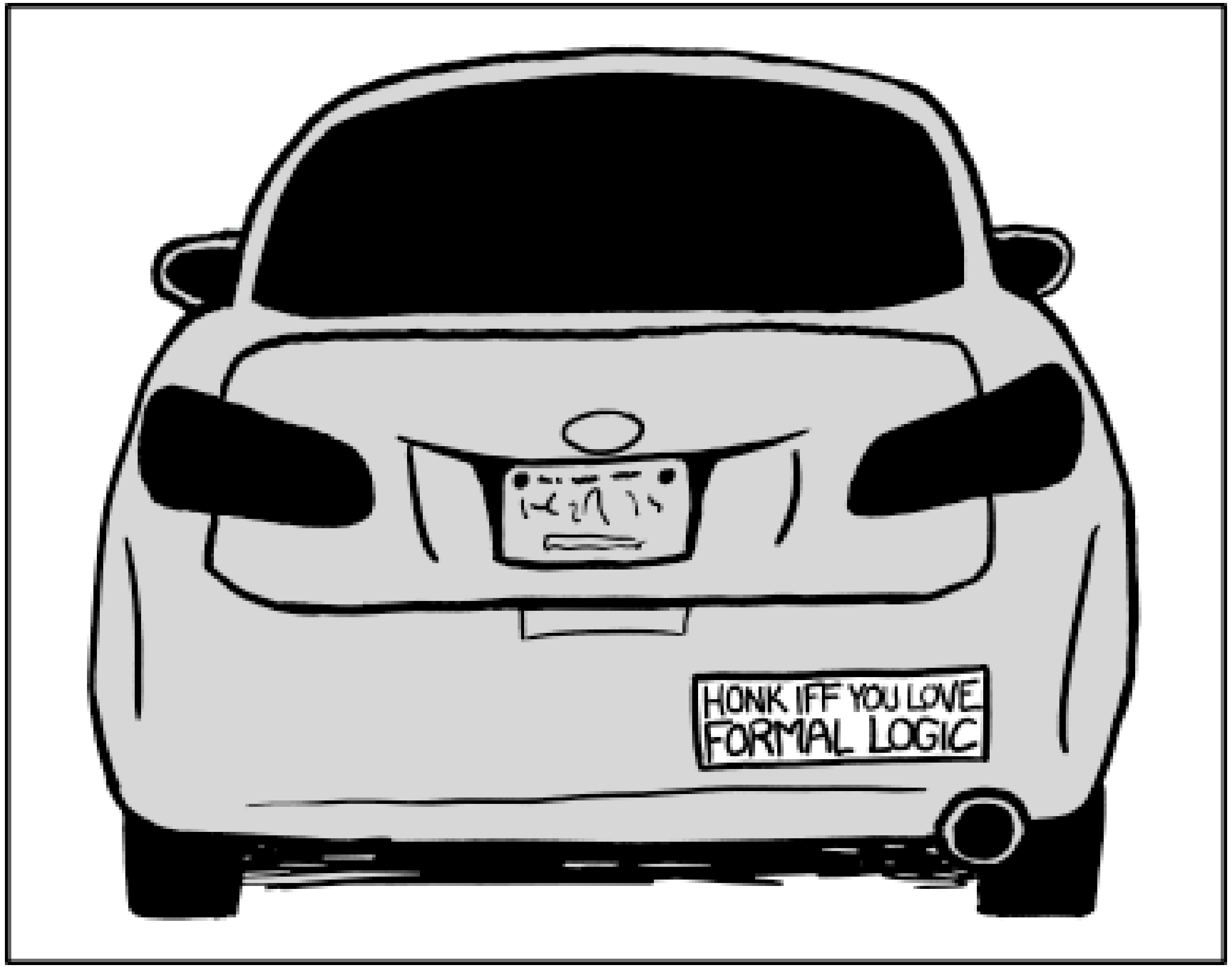
- Domain: $A = \{x \in \mathbb{N} | 0 \leq x \leq 1023\}$

  10 bit binary representation $x_9 x_8 \ldots . x_0$

- $B = \{x \in A | x < 512\}, b = \neg x_9$
- $C = \{x \in A | 256 \leq x < 768\}, c = (\neg x_9 \wedge x_8) \vee (x_9 \wedge \neg x_8)?$

- $D = B \cup C \quad d = \neg x_9 \vee \left((\neg x_9 \wedge x_8) \vee (x_9 \wedge \neg x_8)\right) = \neg x_9 \vee (x_9 \wedge \neg x_8)$
- $E = B \cap C \quad e = \neg x_9 \wedge \left((\neg x_9 \wedge x_8) \vee (x_9 \wedge \neg x_8)\right) = \neg x_9 \wedge x_8$
- $F = A \mid E \quad f = T \wedge \neg(\neg x_9 \wedge x_8) = x_9 \vee \neg x_8$

Thank You



https://xkcd.com/1033/