AI, FAST AND SLOW

Martina Seidl
Institute for Symbolic Artificial Intelligence
LIT AI Lab
Johannes Kepler University Linz

JOHANNES KEPLER
UNIVERSITY LINZ

PLAYING WITH SYMBOLS

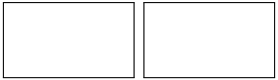
The Box Game

Board:

The Box Game

Board:

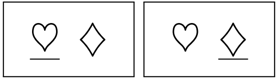
- The board consists of **boxes**.



The Box Game

Board:

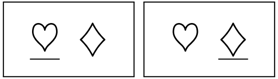
- The board consists of boxes.
- The boxes contain **symbols**.



The Box Game

Board:

- The board consists of boxes.
- The boxes contain symbols.
- Some symbols are **underlined**.

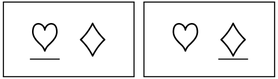


The Box Game

Board:

- The board consists of boxes.
- The boxes contain symbols.
- Some symbols are underlined.

Rules:



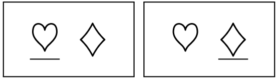
The Box Game

Board:

- The board consists of boxes.
- The boxes contain symbols.
- Some symbols are underlined.

Rules:

- There are 2 colors, e.g., blue and red.



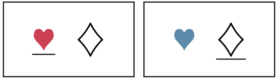
The Box Game

Board:

- The board consists of boxes.
- The boxes contain symbols.
- Some symbols are underlined.

Rules:

- There are 2 colors, e.g., blue and red.
- Task: assign a color to each symbol such that the underlined and non-underlined occurrences of each symbol are colored differently.



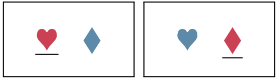
The Box Game

Board:

- The board consists of boxes.
- The boxes contain symbols.
- Some symbols are underlined.

Rules:

- There are 2 colors, e.g., blue and red.
- Task: assign a color to each symbol such that the underlined and non-underlined occurrences of each symbol are colored differently.



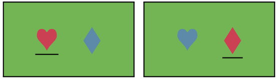
The Box Game

Board:

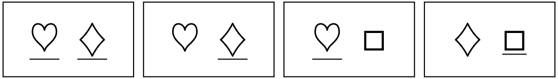
- The board consists of boxes.
- The boxes contain symbols.
- Some symbols are underlined.

Rules:

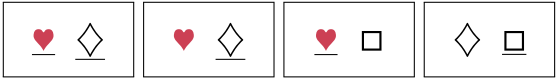
- There are 2 colors, e.g., blue and red.
- Task: assign a color to each symbol such that the underlined and non-underlined occurrences of each symbol are colored differently.
- You win the game if you find a coloring such that each box contains **at least** one symbol in red.



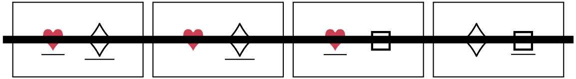
Example



Example

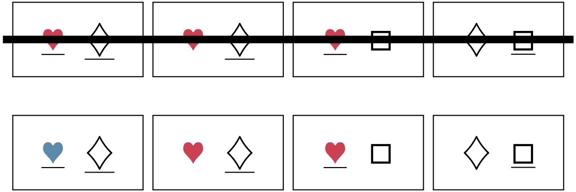


Example



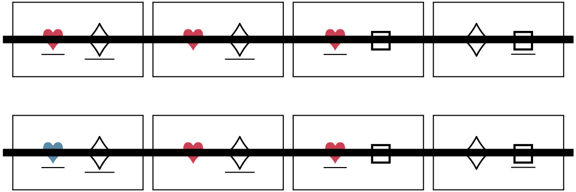
Illegal move!

Example



Illegal move!

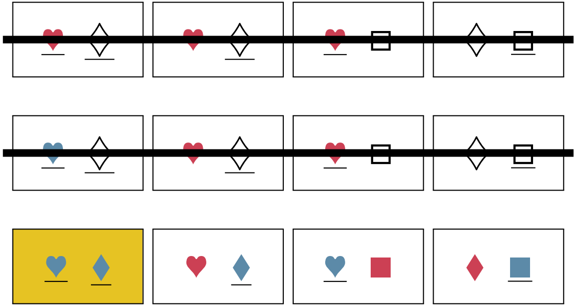
Example



Illegal move!

Illegal move!

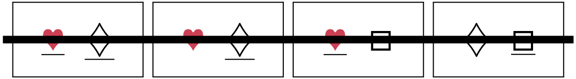
Example



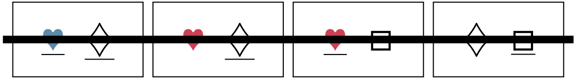
Illegal move!

Illegal move!

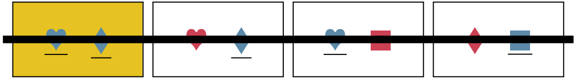
Example



Illegal move!

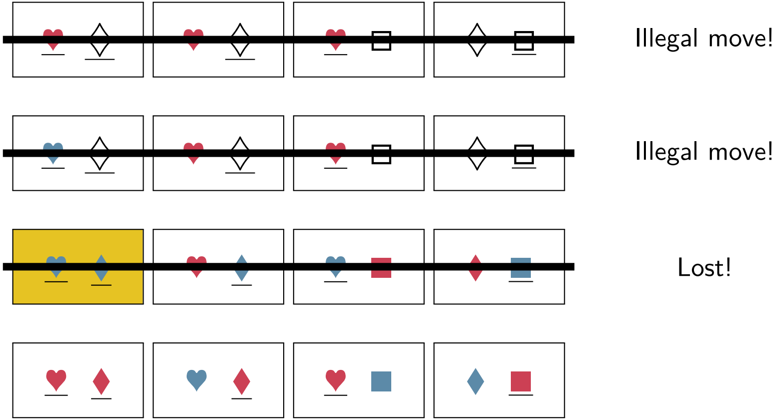


Illegal move!

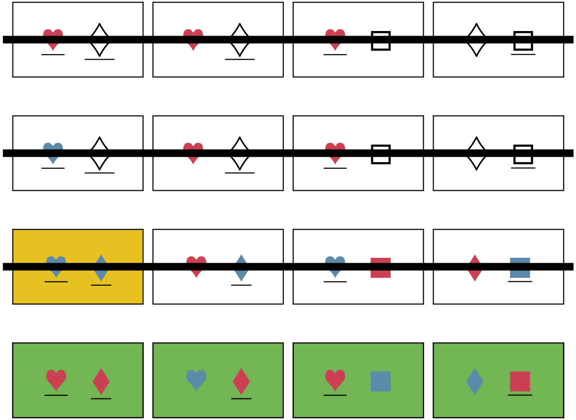


Lost!

Example



Example

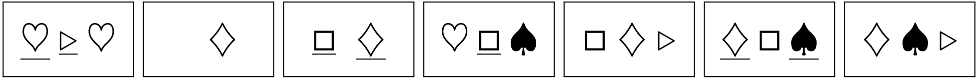


Illegal move!

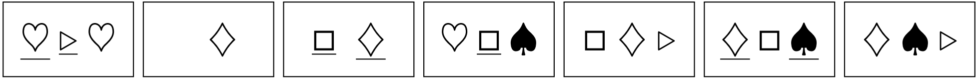
Illegal move!

Lost!

Another Example



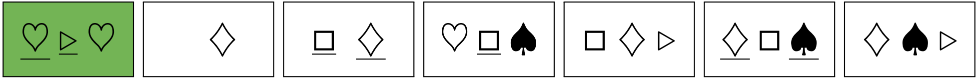
Another Example



Observation 1: Box 1 contains $\underline{\heartsuit}$ and $\heartsuit$.

$\rightarrow$ Box 1 can safely be ignored:
$\heartsuit$ or $\underline{\heartsuit}$ will be assigned in color red.

Another Example

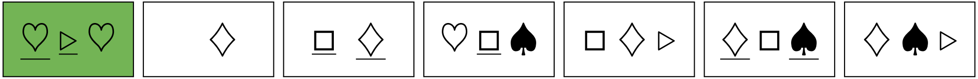


Observation 1: Box 1 contains $\underline{\heartsuit}$ and $\heartsuit$.

$\rightarrow$ Box 1 can safely be ignored:
$\heartsuit$ or $\underline{\heartsuit}$ will be assigned in color red.

Elimination of a tautology

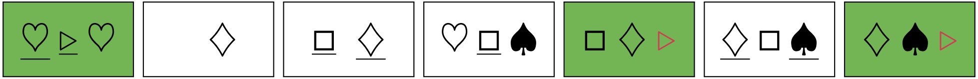
Another Example



Observation 2: $\underline{\triangleright}$ does not occur any more.

$\rightarrow$ Symbol $\triangleright$ can safely be set to red.
(best choice for $\triangleright$)

Another Example

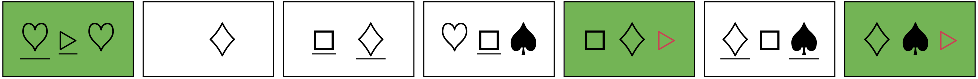


Observation 2: $\underline{\triangleright}$ does not occur any more.

$\rightarrow$ Symbol $\triangleright$ can safely be set to red.
(best choice for $\triangleright$)

Pure-Literal Rule

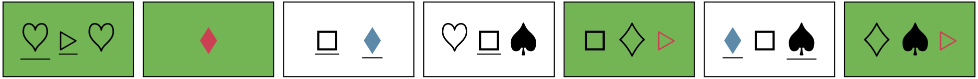
Another Example



Observation 3a: Box 2 contains only one symbol: $\diamondsuit$.

$\rightarrow$ $\diamondsuit$ needs to be red,
otherwise the game is lost.

Another Example

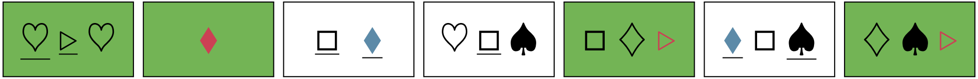


Observation 3a: Box 2 contains only one symbol: $\diamondsuit$.

$\rightarrow$ $\diamondsuit$ needs to be red,
otherwise the game is lost.

Unit Propagation

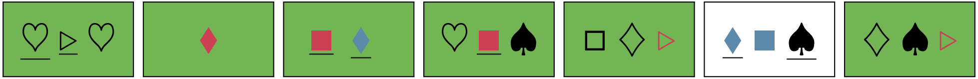
Another Example



Observation 3b: Box 3 contains only one symbol: $\underline{\square}$.

$\rightarrow$ $\underline{\square}$ must be red,
otherwise the game is lost.

Another Example

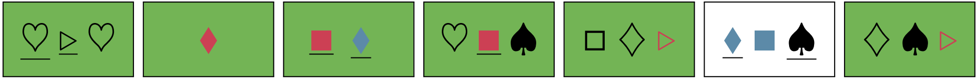


Observation 3b: Box 3 contains only one symbol: □.

→ □ must be red,
otherwise the game is lost.

Unit Propagation

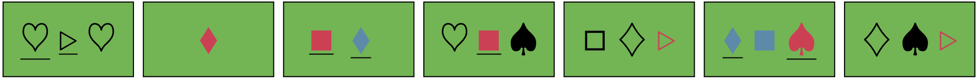
Another Example



Observation 3c: Box 6 contains only one symbol: ♠.

→ ♠ must be red,
otherwise the game is lost

Another Example

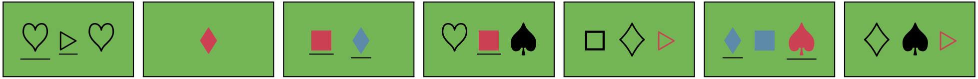


Observation 3c: Box 6 contains only one symbol: ♠.

→ ♠ must be red,
otherwise the game is lost

Unit Propagierung

Another Example



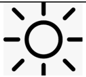
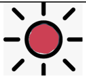
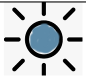
We won!!!!

Information über die Struktur des Problems hat uns beim Lösen geholfen!

The Rubber-Boots Problem

Encoding & Solution

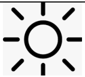
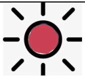
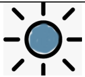
Meaning of Symbols:

	?	yes	no
rubber boots			
shoes			
sun			

Rules:

Encoding & Solution

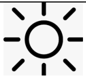
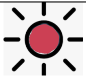
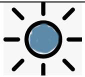
Meaning of Symbols:

	?	yes	no
rubber boots			
shoes			
sun			

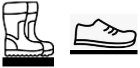
Rules:

Encoding & Solution

Meaning of Symbols:

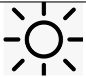
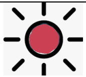
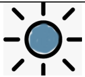
	?	yes	no
rubber boots			
shoes			
sun			

Rules:

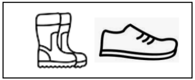


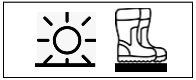
Encoding & Solution

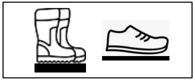
Meaning of Symbols:

	?	yes	no
rubber boots			
shoes			
sun			

Rules:

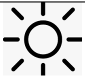
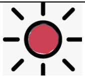
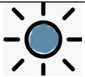




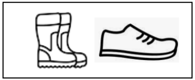


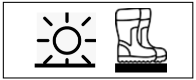
Encoding & Solution

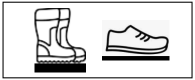
Meaning of Symbols:

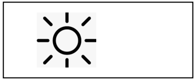
	?	yes	no
rubber boots			
shoes			
sun			

Rules:



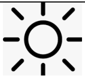
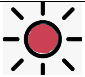
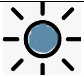




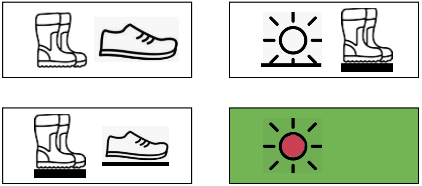


Encoding & Solution

Meaning of Symbols:

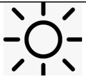
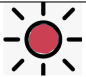
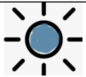
	?	yes	no
rubber boots			
shoes			
sun			

Rules:

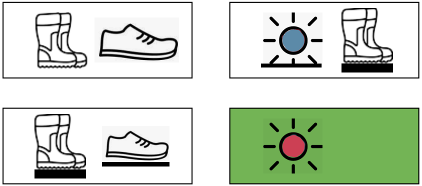


Encoding & Solution

Meaning of Symbols:

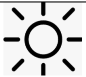
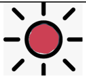
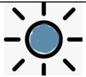
	?	yes	no
rubber boots			
shoes			
sun			

Rules:

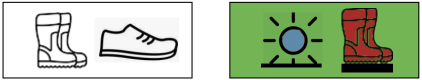


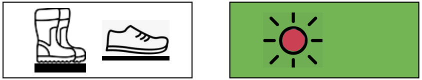
Encoding & Solution

Meaning of Symbols:

	?	yes	no
rubber boots			
shoes			
sun			

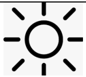
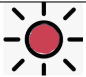
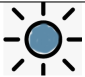
Rules:



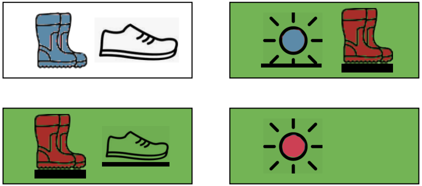


Encoding & Solution

Meaning of Symbols:

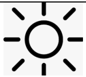
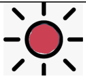
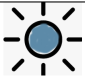
	?	yes	no
rubber boots			
shoes			
sun			

Rules:

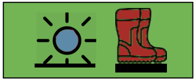


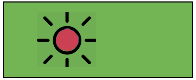
Encoding & Solution

Meaning of Symbols:

	?	yes	no
rubber boots			
shoes			
sun			

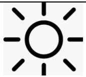
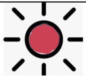
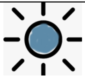
Rules:



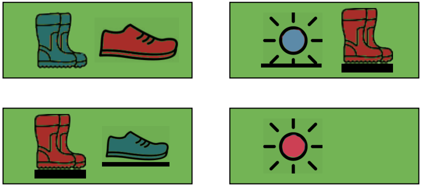


Encoding & Solution

Meaning of Symbols:

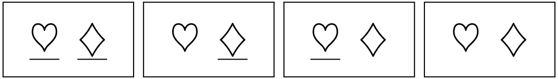
	?	yes	no
rubber boots			
shoes			
sun			

Rules:

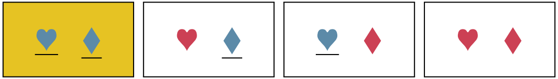


⇒ **Solution: Wear shoes!**

Another Example!

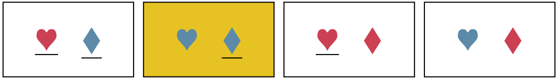


Another Example!



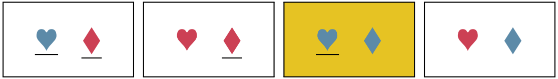
- ♥, ♦: Lost !

Another Example!



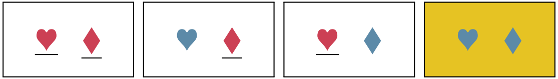
- $\heartsuit$, $\diamondsuit$: Lost !
- $\underline{\heartsuit}$, $\diamondsuit$: Lost !

Another Example!



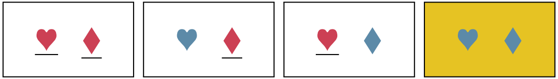
- $\heartsuit$, $\diamondsuit$: Lost !
- $\underline{\heartsuit}$, $\diamondsuit$: Lost !
- $\heartsuit$, $\underline{\diamondsuit}$: Lost !

Another Example!



- ♥, ♦: Lost !
- $\underline{\heartsuit}$, ♦: Lost !
- ♥, $\underline{\diamondsuit}$: Lost !
- $\underline{\heartsuit}$, $\underline{\diamondsuit}$: Lost !

Another Example!



- ♥, ♦: Lost !
- $\underline{\heartsuit}$, ♦: Lost !
- ♥, $\underline{\diamondsuit}$: Lost !
- $\underline{\heartsuit}$, $\underline{\diamondsuit}$: Lost !

There is no solution !!!!!!

Number of Solution Candidates

- 1 symbol, 2 possibilities
  1. ♥
  2. ♥

Number of Solution Candidates

- 1 symbol, 2 possibilities
  1. ♥
  2. ♥
- 2 symbols, 4 possibilities
  1. ♥, ♦
  2. ♥, ♦
  3. ♥, ♦
  4. ♥, ♦

Number of Solution Candidates

- 1 symbol, 2 possibilities
  1. ♥
  2. ♥
- 2 symbols, 4 possibilities
  1. ♥, ♦
  2. ♥, ♦
  3. ♥, ♦
  4. ♥, ♦
- 3 symbols, 8 possibilities
  1. ♥, ♦, ■
  2. ♥, ♦, ■
  3. ♥, ♦, ■
  4. ♥, ♦, ■
  5. ♥, ♦, ■
  6. ♥, ♦, ■
  7. ♥, ♦, ■
  8. ♥, ♦, ■

Number of Solution Candidates

- 1 symbol, 2 possibilities
  1. $\color{red}\heartsuit$
  2. $\color{blue}\heartsuit$
- 2 symbols, 4 possibilities

1.	$\color{red}\heartsuit$, $\color{red}\diamondsuit$	2.	$\color{red}\heartsuit$, $\color{blue}\diamondsuit$
3.	$\color{blue}\heartsuit$, $\color{red}\diamondsuit$	4.	$\color{blue}\heartsuit$, $\color{blue}\diamondsuit$

- 3 symbols, 8 possibilities

1.	$\color{red}\heartsuit$, $\color{red}\diamondsuit$, $\color{red}\blacksquare$	2.	$\color{red}\heartsuit$, $\color{blue}\diamondsuit$, $\color{red}\blacksquare$
3.	$\color{blue}\heartsuit$, $\color{red}\diamondsuit$, $\color{red}\blacksquare$	4.	$\color{blue}\heartsuit$, $\color{blue}\diamondsuit$, $\color{red}\blacksquare$
5.	$\color{red}\heartsuit$, $\color{red}\diamondsuit$, $\color{blue}\blacksquare$	6.	$\color{red}\heartsuit$, $\color{blue}\diamondsuit$, $\color{blue}\blacksquare$
7.	$\color{blue}\heartsuit$, $\color{red}\diamondsuit$, $\color{blue}\blacksquare$	8.	$\color{blue}\heartsuit$, $\color{blue}\diamondsuit$, $\color{blue}\blacksquare$

- 4 symbols, 16 possibilities

  ...

Number of Solution Candidates

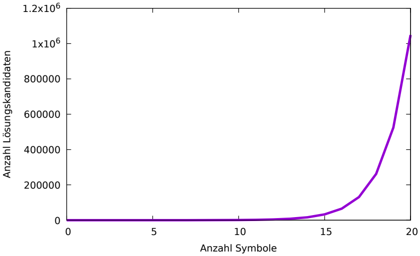
- 1 symbol, 2 possibilities
  1. ♥
  2. ♥
- 2 symbols, 4 possibilities
  1. ♥, ♦ 2. ♥, ♦
  3. ♥, ♦ 4. ♥, ♦
- 3 symbols, 8 possibilities
  1. ♥, ♦, ■ 2. ♥, ♦, ■
  3. ♥, ♦, ■ 4. ♥, ♦, ■
  5. ♥, ♦, ■ 6. ♥, ♦, ■
  7. ♥, ♦, ■ 8. ♥, ♦, ■
- 4 symbols, 16 possibilities

  ...

n symbols

$\rightarrow$

2^n possibilities



Now What?

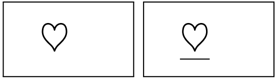
Now What?

- **Observation 1**: If the game contains the empty box, there is no solution.

Now What?

- **Observation 1**: If the game contains the empty box, there is no solution.

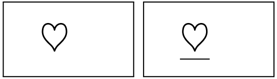
- **Observation 2**: Also in this case there is no solution:



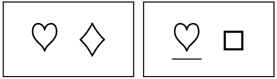
Now What?

- **Observation 1**: If the game contains the empty box, there is no solution.

- **Observation 2**: Also in this case there is no solution:



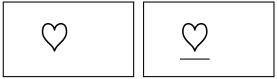
- **Idea:** Find a rule to safely derive the empty box.



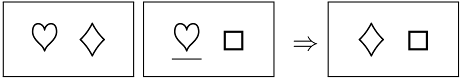
Now What?

- **Observation 1**: If the game contains the empty box, there is no solution.

- **Observation 2**: Also in this case there is no solution:



- **Idea:** Find a rule to safely derive the empty box.



Example

1: $\boxed{\underline{\heartsuit} \quad \underline{\diamondsuit}}$ 2: $\boxed{\heartsuit \quad \underline{\diamondsuit}}$ 3: $\boxed{\underline{\heartsuit} \quad \diamondsuit}$ 4: $\boxed{\heartsuit \quad \diamondsuit}$

Example

1: $\boxed{\underline{\heartsuit}\quad\underline{\diamondsuit}}$ 2: $\boxed{\heartsuit\quad\underline{\diamondsuit}}$ 3: $\boxed{\underline{\heartsuit}\quad\diamondsuit}$ 4: $\boxed{\heartsuit\quad\diamondsuit}$

$1 + 2 \quad \Rightarrow \quad$ 5: $\boxed{\underline{\diamondsuit}\qquad}$

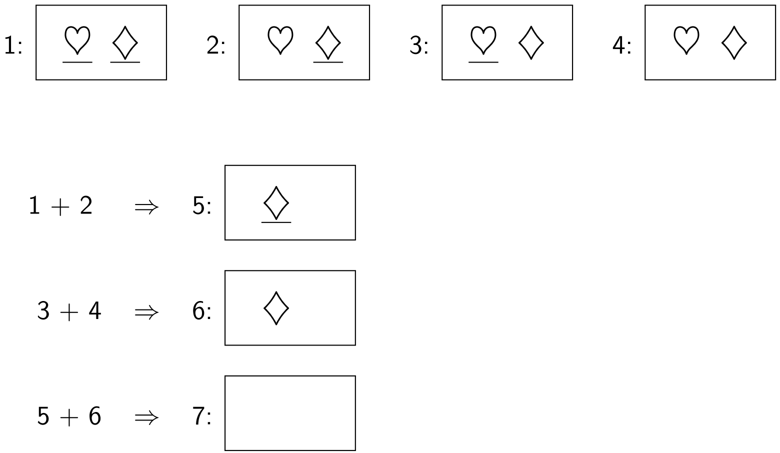
Example

1: $\underline{\heartsuit}\quad\underline{\diamondsuit}$ 2: $\heartsuit\quad\underline{\diamondsuit}$ 3: $\underline{\heartsuit}\quad\diamondsuit$ 4: $\heartsuit\quad\diamondsuit$

$1 + 2 \quad\Rightarrow\quad$ 5: $\underline{\diamondsuit}$

$3 + 4 \quad\Rightarrow\quad$ 6: $\diamondsuit$

Example



Summary

Properties of the Box Game

Summary

Properties of the Box Game

1. Solutions are easy to check.

Summary

Properties of the Box Game

1. Solutions are easy to check.
2. Finding solutions is hard.

Summary

Properties of the Box Game

1. Solutions are easy to check.
2. Finding solutions is hard.

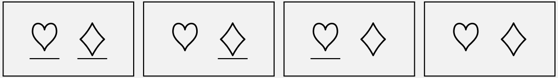
Does this remind us of something????

Summary

Properties of the Box Game

1. Solutions are easy to check.
2. Finding solutions is hard.

Does this remind us of something????

Box Game: 

corresponds to

propositional formula: $(\neg a \vee \neg b) \wedge (a \vee \neg b) \wedge (\neg a \vee b) \wedge (a \vee b)$

SAT SOLVING

Propositional Logic: Syntax and Semantics

Elements of a formula:

- **literal**: variable or negated variable
- **clause**: disjunction of literals
- **formula in CNF (conjunctive normal form)**: conjunction of clauses

Example

$$(\neg u \lor z) \land (y \lor u \lor \neg z) \land (x \lor \neg u \lor \neg z)$$

Propositional Logic: Syntax and Semantics

Elements of a formula:

- **literal**: variable or negated variable
- **clause**: disjunction of literals
- **formula in CNF (conjunctive normal form)**: conjunction of clauses

Example

$$(\neg u \vee z) \wedge (y \vee u \vee \neg z) \wedge (x \vee \neg u \vee \neg z)$$

Semantics: A CNF formula is true under an assignment σ of the Boolean variables iff each clause contains at least one literal that is true under σ.

Core Technique I: Resolution

Proof system with two rules:

Clause Axiom

$$\frac{}{C} \qquad \text{(cl-init)}$$

Core Technique I: Resolution

Proof system with two rules:

Clause Axiom

$$\frac{}{\mathsf{C}} \qquad \text{(cl-init)}$$

Resolution Rule

$$\frac{C_1 \vee p \qquad C_2 \vee \bar{p}}{C_1 \vee C_2} \qquad (res)$$

Core Technique I: Resolution

Proof system with two rules:

Clause Axiom

$$\frac{}{\mathsf{C}} \qquad \text{(cl-init)}$$

Resolution Rule

$$\frac{C_1 \vee p \qquad C_2 \vee \bar{p}}{C_1 \vee C_2} \qquad (res)$$

- in other words:
  $(\neg p \to C_1)$ AND $(p \to C_2)$ DERIVE $C_1 \vee C_2$

Core Technique I: Resolution

Proof system with two rules:

Clause Axiom

$$\frac{}{\mathsf{C}} \qquad \text{(cl-init)}$$

Resolution Rule

$$\frac{C_1 \vee p \qquad C_2 \vee \bar{p}}{C_1 \vee C_2} \qquad (res)$$

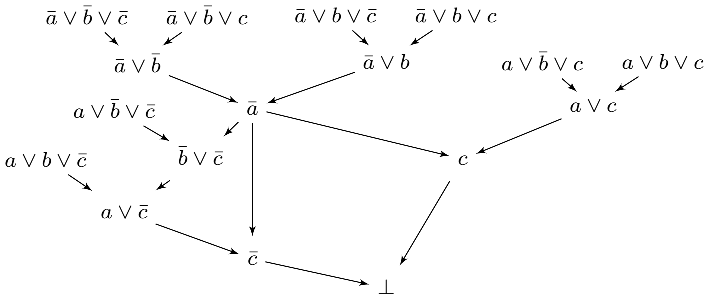
- in other words:
  $(\neg p \to C_1)$ AND $(p \to C_2)$ DERIVE $C_1 \vee C_2$
- resolution is **sound** and **complete**

J. A. Robinson. 1965. A Machine-Oriented Logic Based on the Resolution Principle. J. ACM 12(1), 23–41

Resolution Example

We **prove unsatisfiability** of

$$(a \vee b \vee c) \wedge (a \vee b \vee \neg c) \wedge (\neg a \vee b \vee c) \wedge (\neg a \vee b \vee \neg c) \wedge (a \vee \neg b \vee c) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee \neg b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$$



by resolution as follows:

Core Technique II: Boolean Constraint Propagation (BCP)

Let ϕ be a formula in CNF containing a unit clause C, i.e., ϕ has a clause $C = (l)$ which consists only of literal l. Then $BCP(\phi, l)$ is obtained from ϕ by

- removing all clauses with l
- removing all occurrences of $\bar{l}$

- BCP can trigger other applications of BCP
- if BCP results in empty clause, then formula is unsatisfiable
- if BCP produces the empty CNF, then formula satisfiable

Example BCP

$$\phi = \{(\neg a \vee b \vee \neg c), (a \vee b), (\neg a \vee \neg b), (a)\}$$

Example BCP

$$\phi = \{(\neg a \vee b \vee \neg c), (a \vee b), (\neg a \vee \neg b), (a)\}$$

1. $\phi' = BCP(\phi, a) = \{(b \vee \neg c), (\neg b)\}$

Example BCP

$$\phi = \{(\neg a \lor b \lor \neg c), (a \lor b), (\neg a \lor \neg b), (a)\}$$

1. $\phi' = BCP(\phi, a) = \{(b \lor \neg c), (\neg b)\}$
2. $\phi'' = BCP(\phi', \neg b) = \{(\neg c)\}$

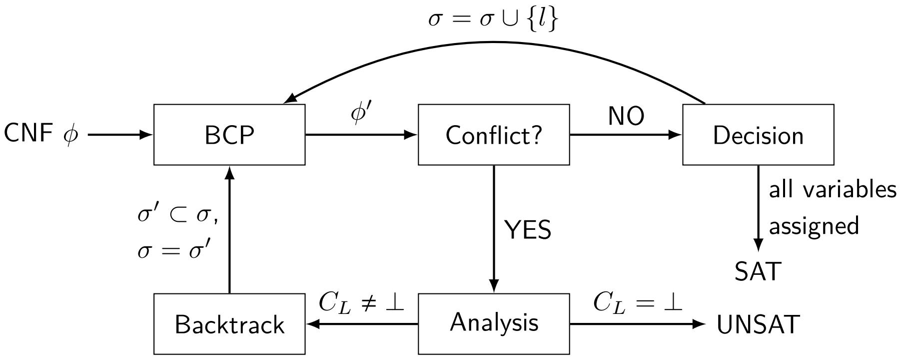
Example BCP

$$\phi = \{(\neg a \vee b \vee \neg c), (a \vee b), (\neg a \vee \neg b), (a)\}$$

1. $\phi' = BCP(\phi, a) = \{(b \vee \neg c), (\neg b)\}$
2. $\phi'' = BCP(\phi', \neg b) = \{(\neg c)\}$
3. $\phi'' = BCP(\phi', c) = \{\} = \top$

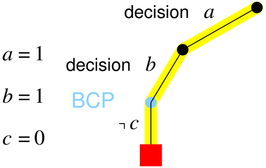
Conflict-Driven Clause Learning (CDCL)

J. Marques-Silva, I. Lynce, S. Malik: Conflict-Driven Clause Learning SAT Solvers. Handbook of SAT 2021



Clause Learning by Example

Based on example by Armin Biere



clauses

$\neg a \vee \neg b \vee \neg c$
$\neg a \vee \neg b \vee c$
$\neg a \vee b \vee \neg c$
$\neg a \vee b \vee c$
$a \vee \neg b \vee \neg c$
$a \vee \neg b \vee c$
$a \vee b \vee \neg c$
$a \vee b \vee c$

learn $\neg a \vee \neg b$

$$\bar{a} \vee \bar{b} \vee \bar{c} \qquad \bar{a} \vee \bar{b} \vee c$$
$$\searrow \quad \swarrow$$
$$\bar{a} \vee \bar{b}$$

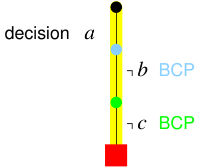
Clause Learning by Example

Based on example by Armin Biere

$a = 1$

$b = 0$

$c = 0$



clauses

$\neg a \vee \neg b \vee \neg c$

$\neg a \vee \neg b \vee c$

$\neg a \vee b \vee \neg c$

$\neg a \vee b \vee c$

$a \vee \neg b \vee \neg c$

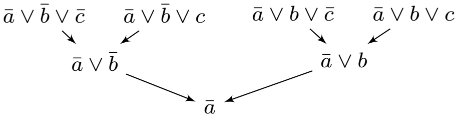
$a \vee \neg b \vee c$

$a \vee b \vee \neg c$

$a \vee b \vee c$

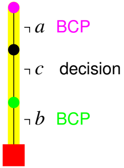
$\neg a \vee \neg b$

learn $\neg a$



Clause Learning by Example

Based on example by Armin Biere



clauses

$\neg a \vee \neg b \vee \neg c$

$\neg a \vee \neg b \vee c$

$\neg a \vee b \vee \neg c$

$\neg a \vee b \vee c$

$a \vee \neg b \vee \neg c$

$a \vee \neg b \vee c$

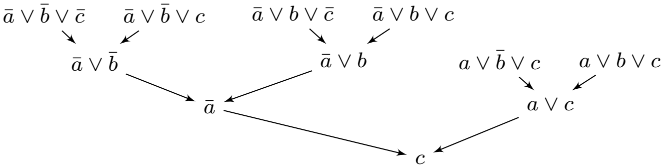
$a \vee b \vee \neg c$

$a \vee b \vee c$

$\neg a \vee \neg b$

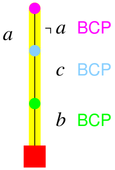
$\neg a$

learn c



Clause Learning by Example

Based on example by Armin Biere



clauses

$\neg a \vee \neg b \vee \neg c$

$\neg a \vee \neg b \vee c$

$\neg a \vee b \vee \neg c$

$\neg a \vee b \vee c$

$a \vee \neg b \vee \neg c$

$a \vee \neg b \vee c$

$a \vee b \vee \neg c$

$a \vee b \vee c$

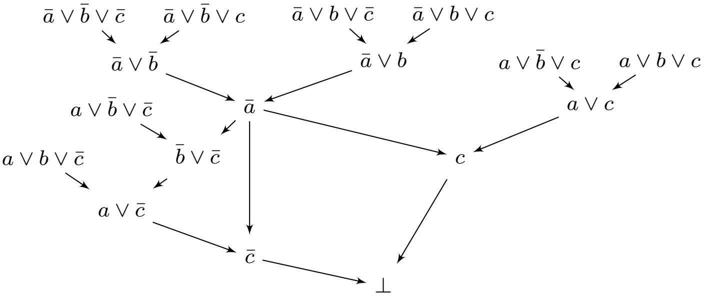
$\neg a \vee \neg b$

$\neg a$

c

learn $\perp$

empty clause



How to Ensure Correctness of SAT Solvers?

1. Carefull testing: incomplete

How to Ensure Correctness of SAT Solvers?

1. Carefull testing: incomplete

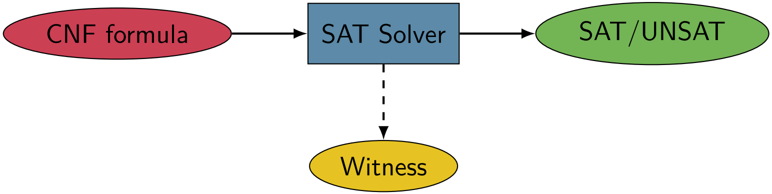
2. Verification of SAT Solver: not feasible in general

How to Ensure Correctness of SAT Solvers?

1. Carefull testing: incomplete
2. Verification of SAT Solver: not feasible in general
3. Check result by independent checker

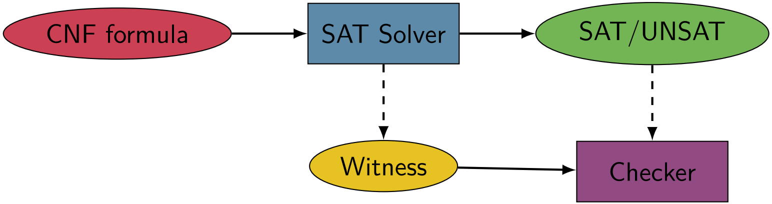
How to Ensure Correctness of SAT Solvers?

1. Carefull testing: incomplete
2. Verification of SAT Solver: not feasible in general
3. Check result by independent checker



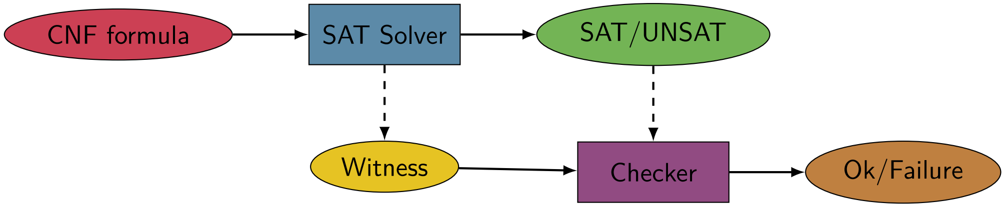
How to Ensure Correctness of SAT Solvers?

1. Carefull testing: incomplete
2. Verification of SAT Solver: not feasible in general
3. Check result by independent checker



How to Ensure Correctness of SAT Solvers?

1. Carefull testing: incomplete
2. Verification of SAT Solver: not feasible in general
3. Check result by independent checker



Witnesses

- True formula: easy
  Check if the assignment returned by SAT solver is a satisfying assignment.

Witnesses

- True formula: easy
  Check if the assignment returned by SAT solver is a satisfying assignment.

- False formula: not so easy, but doable
  - unsatisfiability proof (resolution, RUP, RAT, ...)
  - ideally, checking is polynomial in the proof size

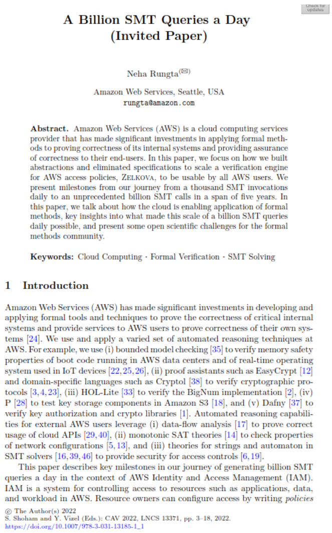
Features of Modern SAT Solvers

- Well-defined interfaces
  - standard format DIMACS
  - API
- Many options
  - carefully selected default configurations
  - highly configurable
- Standardized proof logging
  - for sat and unsat formulas
  - efficient and verified checkers
- Incremental solving
- Parallel and distributed solving
- Unsat core extraction
- ...

PRACTICAL APPLICATIONS OF SYMBOLIC REASONING

Symbolic Techniques for Better Software and Hardware

- **Correctness checking**
  Does the implementation follow the specification?
- **Equivalence checking**
  Does an optimization change the behavior of the system?
- **Synthesis**
  Automatic generation of implementation from specification
- **Test case generation**
- …



A Billion SMT Queries a Day
(Invited Paper)

Neha Rungta(✉)

Amazon Web Services, Seattle, USA
rungta@amazon.com

Abstract. Amazon Web Services (AWS) is a cloud computing services provider that has made significant investments in applying formal methods to proving correctness of its internal systems and providing assurance of correctness to their end-users. In this paper, we focus on how we built abstractions and eliminated specifications to scale a verification engine for AWS access policies, Zelkova, to be usable by all AWS users. We present milestones from our journey from a thousand SMT invocations daily to an unprecedented billion SMT calls in a span of five years. In this paper, we talk about how the cloud is enabling application of formal methods, key insights into what made this scale of a billion SMT queries daily possible, and present some open scientific challenges for the formal methods community.

Keywords: Cloud Computing · Formal Verification · SMT Solving

1 Introduction

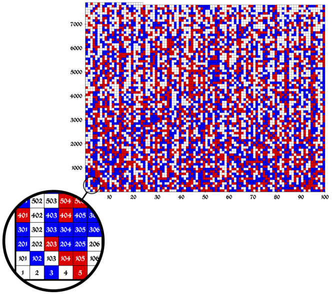
Amazon Web Services (AWS) has made significant investments in developing and applying formal tools and techniques to prove the correctness of critical internal systems and provide services to AWS users to prove correctness of their own systems [24]. We use and apply a varied set of automated reasoning techniques at AWS. For example, we use (i) bounded model checking [35] to verify memory safety properties of boot code running in AWS data centers and of real-time operating system used in IoT devices [22,25,26], (ii) proof assistants such as EasyCrypt [12] and domain-specific languages such as Cryptol [38] to verify cryptographic protocols [3,4,23], (iii) HOL-Lite [33] to verify the BigNum implementation [2], (iv) P [28] to test key storage components in Amazon S3 [18], and (v) Dafny [37] to verify key authorization and crypto libraries [1]. Automated reasoning capabilities for external AWS users leverage (i) data-flow analysis [17] to prove correct usage of cloud APIs [29,40], (ii) monotonic SAT theories [14] to check properties of network configurations [5,13], and (iii) theories for strings and automaton in SMT solvers [16,39,46] to provide security for access controls [6,19].

This paper describes key milestones in our journey of generating billion SMT queries a day in the context of AWS Identity and Access Management (IAM). IAM is a system for controlling access to resources such as applications, data, and workload in AWS. Resource owners can configure access by writing *policies*

© The Author(s) 2022
S. Shoham and Y. Vizel (Eds.): CAV 2022, LNCS 13371, pp. 3–18, 2022.
https://doi.org/10.1007/978-3-031-13185-1_1

Solving Mathematical Problems

$$a^2 + b^2 = c^2$$



M.Heule et al.: Solving and Verifying the Boolean Pythagorean Triples Problem via Cube-and-Conquer. SAT 2016

nature

Explore content ⌄ About the journal ⌄ Publish with us ⌄

nature > news > article

News | Published: 26 May 2016

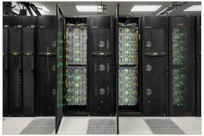
Two-hundred-terabyte maths proof is largest ever

Evelyn Lamb

Nature **534**, 17–18 (2016) | Cite this article

11k Accesses | **7** Citations | **928** Altmetric | Metrics

A computer cracks the Boolean Pythagorean triples problem – but is it really maths?



The University of Texas's Stampede supercomputer, on which the 200-terabyte maths proof was solved. Credit: University of Texas

Matrix Multiplication

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

Matrix Multiplication

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

Standard Matrix Multiplication

$$\begin{array}{ccccc} A_{11} * B_{11} & + & A_{12} * B_{21} & = & C_{11} \\ A_{11} * B_{12} & + & A_{12} * B_{22} & = & C_{12} \\ A_{21} * B_{11} & + & A_{22} * B_{21} & = & C_{21} \\ A_{21} * B_{12} & + & A_{22} * B_{22} & = & C_{22} \end{array}$$

Matrix Multiplication

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

Standard Matrix Multiplication

$$\begin{array}{lcccc} A_{11} * B_{11} & + & A_{12} * B_{21} & = & C_{11} \\ A_{11} * B_{12} & + & A_{12} * B_{22} & = & C_{12} \\ A_{21} * B_{11} & + & A_{22} * B_{21} & = & C_{21} \\ A_{21} * B_{12} & + & A_{22} * B_{22} & = & C_{22} \end{array}$$

Fast Matrix Multiplication

$$\begin{array}{lcccc} (A_{11} + A_{22}) & * & (B_{11} + B_{22}) & = & M_1 \\ (A_{21} + A_{22}) & * & B_{11} & = & M_2 \\ A_{11} & * & (B_{12} - B_{22}) & = & M_3 \\ A_{22} & * & (B_{21} - B_{11}) & = & M_4 \\ (A_{11} + A_{12}) & * & B_{22} & = & M_5 \\ (A_{21} - A_{11}) & * & (B_{11} + B_{12}) & = & M_6 \\ (A_{12} - A_{22}) & * & (B_{21} + B_{22}) & = & M_7 \end{array}$$

$$\begin{array}{lcc} M_1 + M_4 - M_5 + M_7 & = & C_{11} \\ M_3 + M_5 & = & C_{12} \\ M_2 + M_4 & = & C_{21} \\ M_1 - M_2 + M_3 + M_6 & = & C_{22} \end{array}$$

Improvements on Matrix Multiplication

nature

Explore content ⌄ About the journal ⌄ Publish with us ⌄

nature > articles > article

Article | Open access | Published: 05 October 2022

Discovering faster matrix multiplication algorithms with reinforcement learning

Alhussein Fawzi ✉, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis & Pushmeet Kohli

Nature **610**, 47–53 (2022) | Cite this article

Abstract

Improving the efficiency of algorithms for fundamental computations can have a widespread impact, as it can affect the overall speed of a large amount of computations. Matrix multiplication is one such primitive task, occurring in many systems—from neural networks to scientific computing routines. The automatic discovery of algorithms using machine

Improvements on Matrix Multiplication

nature

Explore content ⌄ About the journal ⌄ Publish with us ⌄

nature > articles > article

Article | Open access | Published: 05 October 2022

Discovering faster matrix multiplication algorithms with reinforcement learning

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis & Pushmeet Kohli

Nature **610**, 47–53 (2022) | Cite this article

Abstract

Improving the efficiency of algorithms for fundamental computations can have a widespread impact, as it can affect the overall speed of a large amount of computations. Matrix multiplication is one such primitive task, occurring in many systems—from neural networks to scientific computing routines. The automatic discovery of algorithms using machine

NewScientist

Sign in | Enter search keywords

News Features Newsletters Podcasts Video Comment Culture Crosswords | This week's magazine

Health Space Physics Technology Environment Mind Humans Life Mathematics Chemistry Earth Society

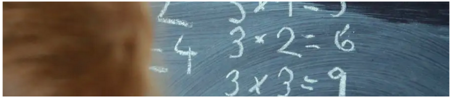
Mathematics

Humans beat DeepMind AI in creating algorithm to multiply numbers

One week after DeepMind revealed an algorithm for multiplying numbers more efficiently, researchers have an even better way to carry out the task

By Matthew Sparkes

13 October 2022



CONCLUSION

Conclusion

Summary

- Symbolic reasoning technology is able to ...
  - ... handle theoretically hard problems
  - ... find solutions where exact answers are required
  - ... provide certified solutions

Conclusion

Summary

- Symbolic reasoning technology is able to ...
  - ... handle theoretically hard problems
  - ... find solutions where exact answers are required
  - ... provide certified solutions

Challenges

- Dealing with (continuous) data / incomplete information
- Currently, deep expert knowledge is required to ...
  - ... find suitable encodings
  - ... build and tune efficient solvers

Conclusion

Summary

- Symbolic reasoning technology is able to ...
  - ... handle theoretically hard problems
  - ... find solutions where exact answers are required
  - ... provide certified solutions

Challenges

- Dealing with (continuous) data / incomplete information
- Currently, deep expert knowledge is required to ...
  - ... find suitable encodings
  - ... build and tune efficient solvers

Here synergies with sub-symbolic techniques could be the solution.

Bilateral
Artificial Intelligence