

# Machine Learning for Theorem Proving

## Lecture 5

Cezary Kaliszyk

March 20, 2018



## Last Lecture: Premise Selection

- SInE
- Regression in theorem proving

## Today

- Decision trees
- Random forests
- Adaptation for prover foundations
- ML-evaluation

# Summary of Last Lecture

## Syntactic methods

- Neighbours using various metrics
- Recursive

MePo, SInE

## Naive Bayes, k-Nearest Neighbours

## Linear / Logistic Regression

- Needs feature and theorem **space reduction**
- Kernel-based multi-output ranking

## Decision Trees (Random Forests)

## Neural Networks

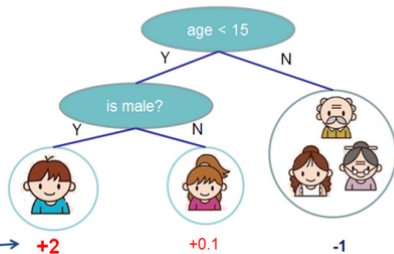
- Winnow, Perceptron
- DeepMath

SNoW, MaLARea

# Decision Trees (1/2)

Input: age, gender, occupation, ...

Does the person like computer games



prediction score in each leaf

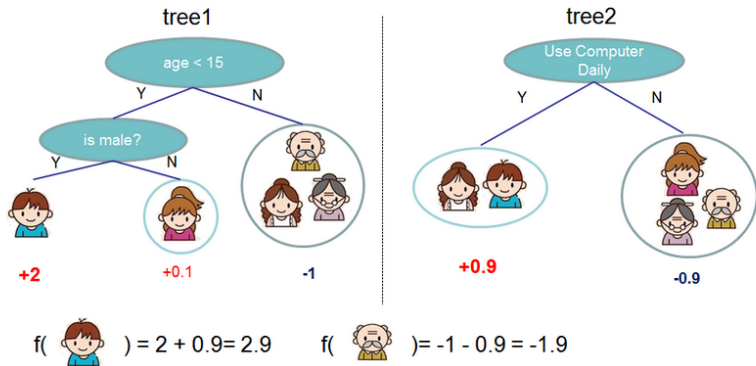
+2

+0.1

-1

[H. Michalewski]

# Decision Trees (2/2)



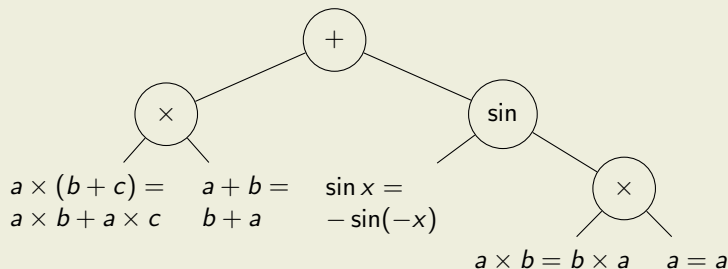
[H. Michalewski]

# Decision Trees

## Definition

- each leaf stores a set of samples
- each branch stores a feature  $f$  and two subtrees, where:
  - the left subtree contains only samples having  $f$
  - the right subtree contains only samples not having  $f$

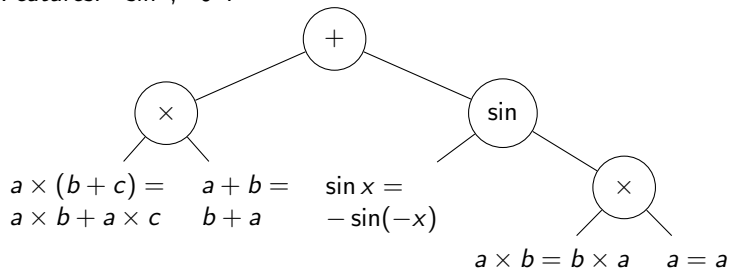
## Example



# Single-path query

Query tree for conjecture " $\sin(0) = 0$ ".

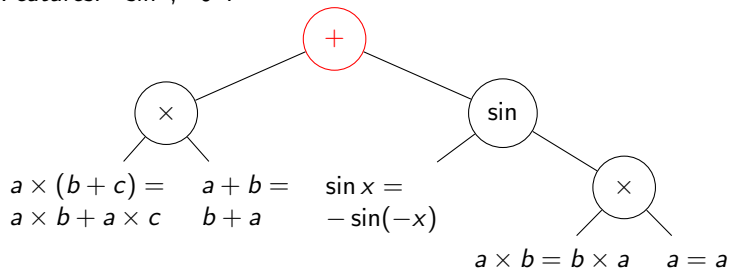
Features: "sin", "0".



# Single-path query

Query tree for conjecture " $\sin(0) = 0$ ".

Features: "sin", "0".

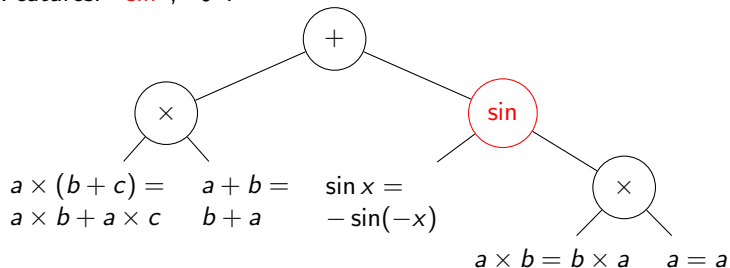




# Single-path query

Query tree for conjecture " $\sin(0) = 0$ ".

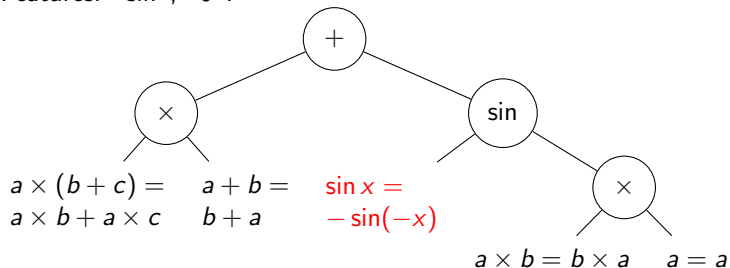
Features: " $\sin$ ", " $0$ ".



# Single-path query

Query tree for conjecture " $\sin(0) = 0$ ".

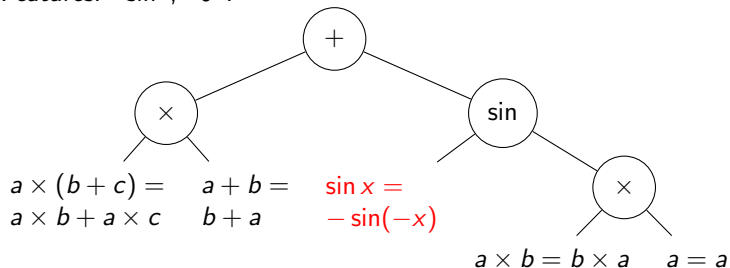
Features: "sin", "0".



# Single-path query

Query tree for conjecture " $\sin(0) = 0$ ".

Features: "sin", "0".

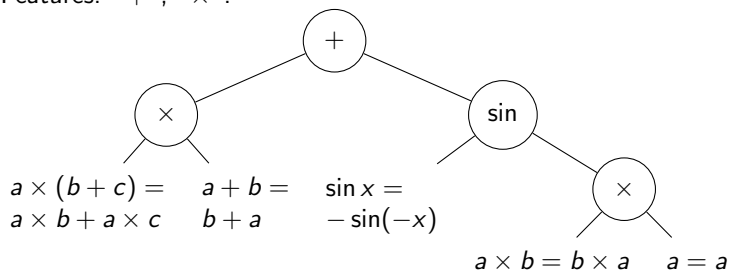


The overall result will be the premises of  $\sin x = -\sin(-x)$ .

## Single-path query (2)

Query tree for conjecture " $(a + b) \times c = a \times c + b \times c$ ".

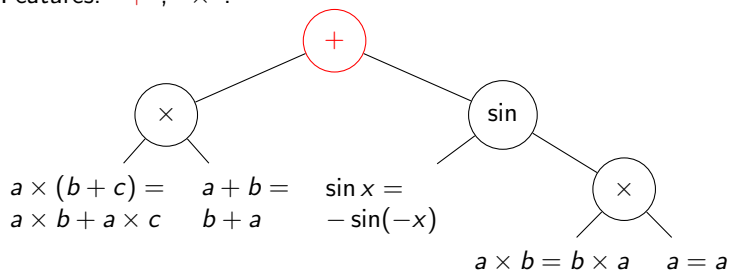
Features: "+", "×".



## Single-path query (2)

Query tree for conjecture " $(a + b) \times c = a \times c + b \times c$ ".

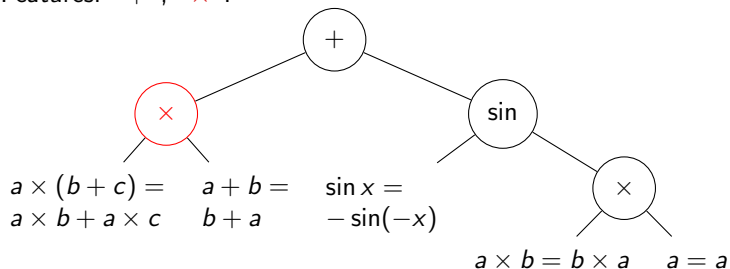
Features: "+", "×".



## Single-path query (2)

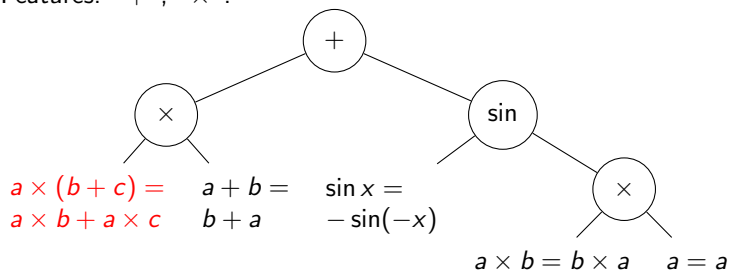
Query tree for conjecture " $(a + b) \times c = a \times c + b \times c$ ".

Features: "+", "×".



## Single-path query (2)

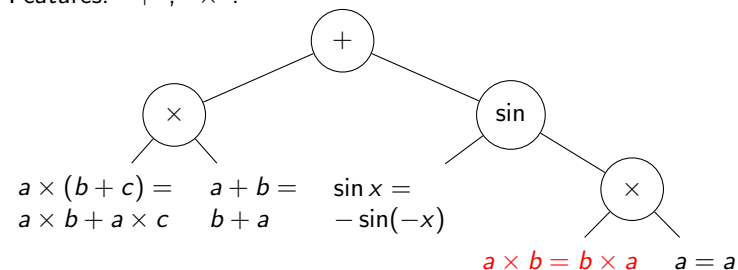
Query tree for conjecture " $(a + b) \times c = a \times c + b \times c$ ".  
Features: "+", "×".



## Single-path query (2)

Query tree for conjecture " $(a + b) \times c = a \times c + b \times c$ ".

Features: "+", "×".



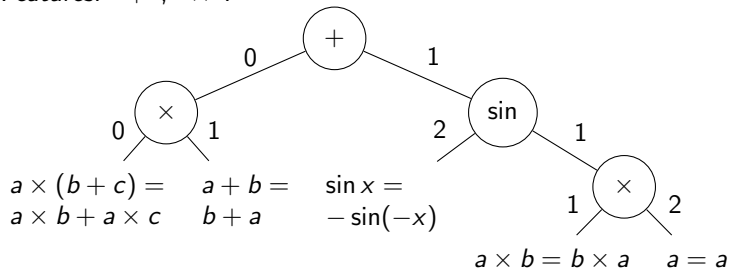
$a \times b = b \times a$  is not considered!



# Multi-path query

Weight samples by the number of errors on each path.

Features: "+", "×".



# Splitting feature

## Agrawal et al.

- Take  $n$  random features from samples and choose feature with lowest Gini impurity (probability of mis-labeling)
- Problem: Gini impurity calculation slow
- Choose feature that divides samples most evenly ( $|S_f| \approx |S_{-f}|$ )

## Online / Offline forests

tree is updated or completely rebuilt

[Agrawal, Saffari]

## Approach for premise selection

- when a branch learns new samples, check whether the branch feature is still an optimal splitting feature wrt. the new data
- if yes, update subtrees with new data
- if no, rebuild tree

learning takes 21 min for the Mizar dataset...

# Different Prover Foundations

FOL, HOL, ...

## Set Theory

- sets and membership
- semantic information
- “collections of things”
- membership is undecidable
- extensional; talk about things that exist

## Type Theory

- typing judgement
- syntactic information
- what objects can be constructed
- intentional
- type checking (and sometimes inference) is decidable

# Typed $\lambda$ -calculus as a foundation

## Basis for a Proof Assistant

- Terms: Programs and Proofs
- Types: Specifications and Formulas

## Brings together

- Programming
- Proving

## Extensions

- Dependent Types
- Dependent Propositions
- Dependency on Proofs
- Reflection ...

## What is a set?

- Sets are commonly used in mathematical texts
- There can not be a strict definition of a basic concept

## Definition (Cantor)

A set, is a gathering into one complete object of clearly distinguished objects in our intuition or thought.

## Materialization of a predicate

Given a predicate  $P(x)$ , instead of talking about all the objects that satisfy it, it is easier to consider only one object

$$\{x|P(x)\}$$

# Formal Set Theory

## Naive set theory

- Consider sets as any other objects
- Consequence: sets of sets
- For example

$$S = \{x \mid x \text{ is a set}\}$$

- Paradoxes for naive set theory

## Russell's paradox

$$S = \{x \mid x \text{ is a set and } x \notin x\}$$

## Solutions

Class theory. **Typed set theory**. New foundations...

- $x$  is of type  $D$  (for example  $\mathbb{N}$ ,  $\mathbb{B}$ ,  $\mathbb{R}$ )
- Every domain is a set (trivially distinguished)
- Not every predicate is a valid one for every  $x$

# Typed set theory

- $x$  comes from a certain domain
  - We should have a unique type for any object
- Proper Frankel operator:

$$\{x : D|P(x)\} \text{ or } \{x \in D|P(x)\}$$

## Definition [membership]

$$P(y) \iff y \in \{x : D|P(x)\}$$

## Notation

$$\{x : A|P(x)\} \text{ means } \{x : D|x \in A \wedge P(x)\}$$

# Features, Labels and Training examples (future)

## In FOL / HOL (also set theory)

- Proved theorems  $\rightarrow$  Training examples
- Dependencies (including theorems)  $\rightarrow$  Labels
- Characteristics of conjectures (eg: constants)  $\rightarrow$  Features

## Type Theory: Types, Terms, Thms merged

- Each defined constant consists of a term and a type
- Features are the constants present in the type
- Dependencies are the constants present in the term

## Approaches

- Do we need to?
- Only Prop is insufficient
- Heuristic



## This Lecture

- adaptation for prover foundations
- decision trees
- random forests
- ML-evaluation

## Next

- kernel methods
- deep learning for proving
- ATP introduction