Another Example
▶ Formula \( F : f^3(a) = a \land f^5(a) = a \land f(a) \neq a \)
▶ Current congruence classes:
\[
\{ \{ a, f^3(a) \}, \{ f(a) \}, \{ f^2(a) \}, \{ f^4(a) \}, \{ f^5(a) \} \}
\]
1. Construct the congruence closure \( \sim \) of
\[
\{ s_1 = t_1, \ldots, s_m = t_m \}
\]
over the subterm set \( S_p \).
2. If \( s_i \sim t_i \) for any \( i \) in \([m + 1, n] \), \( F \) is unsatisfiable
3. Otherwise, \( F \) is satisfiable

Another Example, cont
▶ Formula \( F : f^3(a) = a \land f^5(a) = a \land f(a) \neq a \)
▶ Current congruence classes:
\[
\{ \{ a, f^3(a) \}, \{ f(a) \}, \{ f^2(a) \}, \{ f^4(a) \}, \{ f^5(a) \} \}
\]
From \( a = f^3(a) \), what can we infer using function congruence? \( f(a) = f^4(a) \) and \( f^2(a) = f^5(a) \)

Resulting congruence classes:
\[
\{ \{ a, f^4(a) \}, \{ f(a), f^4(a) \}, \{ f^2(a), f^5(a) \} \}
\]

Another Example, cont
▶ Formula \( F : f^3(a) = a \land f^5(a) = a \land f(a) \neq a \)
▶ Current congruence classes:
\[
\{ \{ a, f^3(a) \}, \{ f(a), f^4(a) \}, \{ f^2(a), f^5(a) \} \}
\]
Now, process equality \( f^3(a) = a \); which classes do we merge?
\[
\{ \{ a, f^3(a) \}, \{ f(a), f^4(a) \}, \{ f^2(a), f^5(a) \} \}
\]
From \( a = f^2(a) \), what can we infer via function congruence? \( f(a) = f^4(a) \)

Thus, merge the two congruence classes:
\[
\{ \{ a, f(a), f^2(a), f^4(a), f^5(a) \} \}
\]
Another Example, cont

- Formula $F : f^3(a) = a \land f^5(a) = a \land f(a) \neq a$
- Currenct congruence classes:
  $\{(a, f(a), f^2(a), f^3(a), f^4(a), f^5(a))\}$
- Is the formula satisfiable? No
- Since $f(a)$ and $a$ are in same congruence class, this contradicts $f(a) \neq a$

One More Example

- Consider formula $F : f(x) = f(y) \land x \neq y$
- What is the subterm set? $\{x, y, f(x), f(y)\}$
- Each subterm starts in its own congruence class:
  $\{\{x\}, \{y\}, \{f(x)\}, \{f(y)\}\}$
- Process equality $f(x) = f(y) \Rightarrow \{\{x\}, \{y\}, \{f(x), f(y)\}\}$
- What new equalities can we infer from congruence? None!
- Is the formula satisfiable? Yes

How to Compute Congruence Closure

- So far, we described how to decide satisfiability using congruence closure
- But we haven’t discussed an algorithm for efficiently computing congruence closure
- Next: Efficient algorithm for computing congruence closure

Representing Subterms

- To compute congruence closure efficiently, we’ll represent the subterm set of the formula as a DAG
  - Each node corresponds to a subterm and has unique id
  - Edges point from function symbol to arguments
  - Question: What subterm does node labeled 1 represent? $f(f(a, b), b)$

Representative of Congruence Class

- To compute congruence closure, we need to merge congruence classes
- To do this efficiently, each congruence class has a representative: When merging two classes, only need to update the representative
  - Thus, for a given subterm, we need to be able to find the representative of its class
- Each subterm contains a pointer that eventually leads to the representative of its congruence class
- In this example, $a, f(a, b), f(f(a, b), b)$ are in same congruence class; $a$ is the representative

Parents of a Subterm

- In addition to efficiently finding representative, also need to efficiently find parents of terms
  - Why? Because if $x_1 = y_1, \ldots, x_k = y_k$, function congruence implies $f(\bar{x}) = f(\bar{y})$
  - Thus, when each $x_i, y_i$ pair is in same congruence class, need to merge congruence classes of their parents $f(\bar{x})$ and $f(\bar{y})$
  - Thus, keep pointer from representative of congruence class to parents of all subterms in the congruence class
Summary of Data Structure
- Represent subterms as a DAG
- Each node in the DAG corresponds to a subterm
- Each node stores its unique id, name of function or variable, and list of argument subterms
- Each node \( n \) has a `find` pointer field that leads to its representative
- The `find` field of a representative points to itself
- Each representative stores the set of `parents` for all subterms in that class
- If a term is not a representative, then its `parents` field is empty

Finding Representative of Congruence Class
- Given a term \( t \), we need to find representative for that term
- If \( t \)’s `find` field points to itself, then \( t \) is the representative of its congruence class
- Otherwise, we follow the chain of `find` references until we find a node \( t' \) that points to itself
- In this case, \( t' \) is \( t \)’s representative

Merging Congruence Classes
- Using this data structure, how do we merge congruence classes of two terms \( t_1 \) and \( t_2 \)?
- First find representatives of \( t_1 \) and \( t_2 \) as described earlier
- Want to make \( \text{Rep}(t_2) \) new representative for merged class
- Thus, change `find` field of \( \text{Rep}(t_1) \) to point to \( \text{Rep}(t_2) \)
- Update parents: add parent terms stored in \( \text{Rep}(t_2) \) to those of \( \text{Rep}(t_1) \), and remove parents stored in \( \text{Rep}(t_1) \)

Processing Equalities
- How do we process an equality \( t_1 = t_2 \)?
- Need to merge equivalence classes of \( t_1 \) and \( t_2 \)
- Might potentially also need to merge \( t_1 \) and \( t_2 \)’s parents due to function congruence
- Given parent \( p_1 \) of \( t_1 \) and \( p_2 \) of \( t_2 \), when do we merge \( p_1 \) and \( p_2 \)’s congruence classes?
- If they have the same function name and all of their arguments are congruent (i.e., have same representative)

Processing Equalities, cont
To process equality \( t_1 = t_2 \):
1. Find representatives of \( t_1 \) and \( t_2 \)
2. Merge equivalence classes
3. Retrieve the set of parents \( P_1, P_2 \) stored in \( \text{Rep}(t_1), \text{Rep}(t_2) \)
4. For each \((p_i, p_j) \in P_1 \times P_2\), if \( p_i \) and \( p_j \) are congruent, process equality \( p_i = p_j \)

Observe: Processing one equality creates new equalities, which in turn might generate other new equalities!

Full Algorithm for Deciding Satisfiability
Algorithm to decide satisfiability of \( T_m \) formula
\[
F : s_1 = t_1 \land \ldots \land s_m = t_m \land s_{m+1} \neq t_{m+1} \land \ldots \land s_n \neq t_n
\]
1. Compute subterms and construct initial DAG (each node’s representative is itself)
2. For each \( i \in [1, m] \), process equality \( s_i = t_i \) as described
3. For each \( i \in [m+1, n] \), check if \( \text{Rep}(s_i) = \text{Rep}(t_i) \)
4. If there exists some \( i \in [m+1, n] \) for which \( \text{Rep}(s_i) = \text{Rep}(t_i) \), return UNSAT
5. If for all \( i \), \( \text{Rep}(s_i) \neq \text{Rep}(t_i) \), return SAT
Example

- Consider formula $F : f(a, b) = a \land f(f(a, b), b) \neq a$
- Subterms: $a, b, f(a, b), f(f(a, b), b)$
- Process equality $f(a, b) = a$
- Are parents $f(a, b)$ and $f(f(a, b), b)$ congruent? Yes
- Process equality $f^2(a) = f(a)$
- Formula unsatisfiable because $f(f(a, b), b)$ and $a$ have same representative!

Example II

- Consider formula: $F : f^3(a) = a \land f^5(a) = a \land f(a) \neq a$
- Initial DAG:
- Process equality $f^2(a) = a$:
- Are parents congruent? Yes
- Process equality $f^4(a) = f(a)$

Example II, cont

- After merging classes:
- Are $f^4(a)$'s and $f(a)$'s parents congruent? Yes
- Process equality $f^4(a) = f^2(a)$

Example II, cont

- Formula: $F : f^3(a) = a \land f^5(a) = a \land f(a) \neq a$
- Process equality $f^4(a) = f(a)$
- Now, parents $f^3(a)$ and $f(a)$ congruent; process equality $f^3(a) = f(a)$

Summary

- Congruence closure algorithm is used for determining satisfiability of $T_a$ formulas (without disjunction)
- Deciding conjunctive $T_a$ formulas is inexpensive ($O(e \log (e))$)
Decision procedure for Theory of Rationals: Overview

- We’ll only consider quantifier free $T_Q$ formulas
- Most common technique for deciding satisfiability in $T_Q$ is Simplex algorithm
- Simplex algorithm developed by Dantzig in 1949 for solving linear programming problems
- Since deciding satisfiability of qff conjunctive formulas is a special case of linear programming, we can use Simplex

The Plan

- Overview of linear programming
- Satisfiability as linear programming
- Simplex algorithm

Linear Programming

- In a linear programming (LP) problem, we have an $m \times n$ matrix $A$, an $m$-dimensional vector $b$, and $n$-dimensional vector $\vec{c}$
- Want to find a solution for $\vec{z}$ maximizing objective function $\vec{c}^T \vec{z}$
- subject to linear inequality constraint $A\vec{z} \leq \vec{b}$
- Very important problem; applications in airline scheduling, transportation, telecommunications, finance, production management, marketing, networking, compilers . . .

Geometric Formulation

- For $m \times n$ matrix $A$, the system $A\vec{z} \leq \vec{b}$ forms a convex polytope in $n$-dimensional space
- Polytope is generalization of polyhedron from 3-dim space to higher dimensional space
- Convexity: For all pairs of points $\vec{v}_1, \vec{v}_2$ and for any $\lambda \in [0, 1]$, the point $\lambda \vec{v}_1 + (1 - \lambda) \vec{v}_2$ also lies in polytope
- Goal of linear programming: Find a point that (i) lies inside the polytope, and (ii) maximizes the value of $\vec{c}^T \vec{z}$

Linear Programming Lingo

- In LP, a value of $\vec{x}$ that satisfies constraints $A\vec{x} \leq \vec{b}$ called feasible solution; otherwise, called infeasible solution
- Example: Maximize $2y - x$ subject to:
  $\begin{align*}
  x + y & \leq 3 \\
  2x - y & \leq -5
  \end{align*}$
- Is $(0, 0)$ a feasible solution? No
- What about $(-2, 1)$? Yes
- For a given solution for $\vec{x}$, the corresponding value of objective function $\vec{c}^T \vec{x}$ called objective value
- What is objective value for $(-2, 1)$? 4

Linear Programming Lingo, cont

- A feasible solution whose objective value is maximum over all feasible solutions called optimal solution
- If a linear program has no feasible solutions, the linear program is infeasible
- If optimal solution is $\infty$, then problem is called unbounded
Geometric Interpretation
- Feasible solution is a point within the polytope
- The linear programming problem is infeasible if the polytope defined by $Ax \leq b$ is empty
- An LP problem is unbounded if the polytope is open in the direction of the objective function
- **Question:** If polytope is not closed, does this mean optimal solution is $\infty$? No!
- Since polytope defined by $Ax \leq b$ is convex, optimal solution for bounded LP problem must lie on exterior boundary

Deciding $T_Q$ as Linear Program
- How do we determine $T_Q$ satisfiability using LP?
  - First, convert $T_Q$ formula to NNF.
  - In this form, every atomic formula is of the form: $a_1 x_1 + \ldots + a_n x_n \geq b \quad (\text{all } \in \{=,\neq,\geq,\leq\})$
  - First, rewrite it as equisat formula containing only $\leq$ and $y > 0$
    - $a^T x \geq c \Rightarrow -a^T x \leq -c$
    - $a^T x < c \Rightarrow a^T x + y \leq c \land y > 0$
    - $a^T x = c \Rightarrow a^T x \leq c \land -a^T x \leq -c$
    - $a^T x \neq c \Rightarrow (a^T x + y \leq c \land y > 0) \lor (-a^T x + y \leq -c \land y > 0)$

Deciding $T_Q$ as Linear Program, cont
- Current formula in NNF and no negations
  - Each atomic formula is one of three forms:
    1. $a_1 x_1 + \ldots + a_n x_n \leq b$
    2. $a_1 x_1 + \ldots + a_n x_n + y \leq \beta$
    3. $y > 0$
- Next, convert to DNF: Formula is satisfiable if any of the clauses satisfiable
- Thus, want to formulate each clause as a linear program

Satisfiability as Linear Programming
- Thus, we can formulate satisfiability of every off conjunctive $T_Q$ formula as a linear programming problem.
- Hence, we’ll focus on how to solve LP problems
- Three popular methods for solving LP problems:
  1. Ellipsoid method (Khachian, 1979)
  2. Interior-point algorithm (Karmarkar, 1984)
  3. Simplex algorithm (Dantzig, 1949)
- Among these, ellipsoid and interior-point method are polynomial-time, but Simplex is worst-case exponential
- Despite this, Simplex remains most popular and performs better for most problems of interest