SAT Solving – Parallel Search

Steffen Hölldobler and Norbert Manthey
International Center for Computational Logic
Technische Universität Dresden
Germany

- Parallel Approaches
- Abstract Description
- High-Level
- Problems

"Logic is everywhere ..."
Parallelization – Warm Up

- If an algorithm has three parts that consume 80%, 10% and 10%
Parallelization – Warm Up

► If an algorithm has three parts that consume 80 %, 10 % and 10 %

► Which task should be parallelized?
Parallelization – Warm Up

- If an algorithm has three parts that consume 80 %, 10 % and 10 %

- Which task should be parallelized?

- What is the ideal speedup for two cores?
Parallelization – Warm Up

- Assume two solvers $S_1$ and $S_2$ solve the formula $F$ independently
- $S_1$ learns $C$, $S_2$ learns $D$
Parallelization – Warm Up

- Assume two solvers $S_1$ and $S_2$ solve the formula $F$ independently

- $S_1$ learns $C$, $S_2$ learns $D$

- If $S_1$ receives $D$, is satisfiability preserved?
Parallelization – Warm Up

► Assume two solvers $S_1$ and $S_2$ solve the formula $F$ independently

► $S_1$ learns $C$, $S_2$ learns $D$

► If $S_1$ receives $D$, is satisfiability preserved?

► If $S_1$ receives $D$, is equivalence preserved?
Parallelization – Warm Up

Run unit propagation on

\[ F = (\neg e \lor f) \land (\neg a \lor e) \land (\neg c \lor d) \land (\neg b \lor c) \land (\neg a \lor b) \land a \]
Parallelization – Warm Up

- Run unit propagation on
  \[ F = (\neg e \lor f) \land (\neg a \lor e) \land (\neg c \lor d) \land (\neg b \lor c) \land (\neg a \lor b) \land a \]

- How would you parallelize?
Parallelization – Warm Up

► Run unit propagation on

\[ F = (\neg e \lor f) \land (\neg a \lor e) \land (\neg c \lor d) \land (\neg b \lor c) \land (\neg a \lor b) \land a \]

► How would you parallelize?

► Can the steps be run in parallel?
Parallelization – Warm Up

Run unit propagation on
\[ F = (\neg e \lor f) \land (\neg a \lor e) \land (\neg c \lor d) \land (\neg b \lor c) \land (\neg a \lor b) \land a \]

How would you parallelize?

Can the steps be run in parallel?

Run unit propagation on
\[ F = (\neg e \lor f) \land (\neg d \lor e) \land (\neg c \lor d) \land (\neg b \lor c) \land (\neg a \lor b) \land a \]
Parallelization – Warm Up

Run unit propagation on
\[ F = (\neg e \lor f) \land (\neg a \lor e) \land (\neg c \lor d) \land (\neg b \lor c) \land (\neg a \lor b) \land a \]

How would you parallelize?

Can the steps be run in parallel?

Run unit propagation on
\[ F = (\neg e \lor f) \land (\neg d \lor e) \land (\neg c \lor d) \land (\neg b \lor c) \land (\neg a \lor b) \land a \]

From a complexity point of view, this is an open question!
Parallelization – Revision

- A sequential algorithm
  - requires time $t_1$ seconds

- A parallel algorithm
  - utilizes $p$ computation units (cores)
  - requires time $t_p$ seconds

- The speedup $S_p = \frac{t_1}{t_p}$
  - A speedup $S_p$ is superlinear, if $S_p > p$

- The efficiency $E_p = \frac{S_p}{p}$

- An algorithm is called scalable,
  if it can solve a given problem faster when additional resources are added
Abstract Visualization
Finding an Exit in a Maze

Some rules

- starting point is located in the left column
- exit is on the right side (if there exists one)
- search decisions can be done only when moving right
- when moving left, use backtracking
Finding an Exit in a Maze

- Some rules
  - starting point is located in the left column
  - exit is on the right side (if there exists one)
  - search decisions can be done only when moving right
  - when moving left, use backtracking

- Parallel approaches: multiple searches, search space splitting
Solve With Multiple Solvers

▶ Use a solver per computing resource
  ▶ Use different heuristics
  ▶ Solvers work independently
Solve With Multiple Solvers

- Learned clauses can be shared
  - All solvers work on the same formula
  - No simplification involved
Solve With Multiple Solvers

Arising questions:

- How scalable is the presented approach?
Solve With Multiple Solvers

- Arising questions:
  - How scalable is the presented approach?
  - What influences scalability?
Solve With Multiple Solvers

Arising questions:

- How scalable is the presented approach?
- What influences scalability?
- How long is clause sharing valid wrt simplification?
Solve With Multiple Solvers

Arising questions:

- How scalable is the presented approach?
- What influences scalability?
- How long is clause sharing valid wrt simplification?
- Is there a good alternative?
Partition Search Space

- Create a partition per computing resource
  - Assign a solver to each partition
  - Solvers work independently
Partition Search Space

- Learned clauses can be shared
Partition Search Space

- Learned clauses can be shared
  - with respect to the partition
  - carefully
Partition Search Space

- Partitions can be re-partitioned
  - Ensure load balancing and applies many resources to hard partitions
  - Possible to use learned clauses of parent partition
High Level Parallelization Approaches

Parallel Portfolio Solvers
Search Space Partitioning Solvers
Parallel Portfolio Solvers

- Solve a formula $F$ with $n$ resources
- Idea: solve $F$ with multiple solvers
Parallel Portfolio Solvers

- Solve a formula $F$ with $n$ resources
- Idea: solve $F$ with multiple solvers
  - With different configurations

Drawbacks:
- Known to not scale well
- A small set of configurations is already robust
- Scalability is independent of formula size
Parallel Portfolio Solvers

- Solve a formula $F$ with $n$ resources
- Idea: solve $F$ with multiple solvers
  - With different configurations
  - With knowledge sharing (learned clauses)

Drawbacks:
- Known to not scale well
- A small set of configurations is already robust
- Scalability is independent of formula size
Parallel Portfolio Solvers

- Solve a formula $F$ with $n$ resources
- Idea: solve $F$ with multiple solvers
  - With different configurations
  - With knowledge sharing (learned clauses)

- Drawbacks:
  - Known to not scale well
  - A small set of configurations is already robust
  - Scalability is independent of formula size
Parallel Portfolio Solvers – Sharing

- Given two solvers $S_1$ and $S_2$, and let $S_1$ learn a clause $C$
- Let $F^1$ and $F^2$ be the working formulas of $S_1$ and $S_2$, respectively
- When is $S_2$ allowed to receive $C$

$F^2 \equiv F^2 \land C$ ⊢ $F^2 \equiv SAT$

The check is done implicitly, as it is too expensive

- No simplification, then all clauses are entailed
- Only clause elimination / model increasing techniques, then sharing preserves equisatisfiability

$F^1 = x$, and $S_2$ applies simplification: $F^2 = x$; $modelInc_F^2 = \emptyset$; $modelDec_F^2 = x$

$S_2$ receives $(x)$ from $S_1$: $F^2 = x \land x$

$S$atisfiable formula is found to be unsatisfiable
Parallel Portfolio Solvers – Sharing

- Given two solvers \( S_1 \) and \( S_2 \), and let \( S_1 \) learn a clause \( C \)
- Let \( F^1 \) and \( F^2 \) be the working formulas of \( S_1 \) and \( S_2 \), respectively
- When is \( S_2 \) allowed to receive \( C \)
  - If \( F^2 \equiv F^2 \land C \)
  - If \( F^2 \equiv_{\text{SAT}} F^2 \land C \)

\( \Box \) No simplification, then all clauses are entailed
\( \Box \) Only clause elimination / model increasing techniques, then sharing preserves equisatisfiability
\( \Box \) Addition of redundant, but not entailed, clauses
\( \Box \) Extra care, otherwise:
  - \( F^1 = x \), and \( S_2 \) applies simplification:
  - \( F^2 = x \); \( \text{modelInc} F^2 = \emptyset \); \( \text{modelDec} F^2 = x \)
  - \( S_2 \) receives \((x)\) from \( S_1 \):
  - Satisfiable formula is found to be unsatisfiable
Parallel Portfolio Solvers – Sharing

- Given two solvers $S_1$ and $S_2$, and let $S_1$ learn a clause $C$
- Let $F^1$ and $F^2$ be the working formulas of $S_1$ and $S_2$, respectively
- When is $S_2$ allowed to receive $C$
  - If $F^2 \equiv F^2 \land C$
  - If $F^2 \equiv_{\text{SAT}} F^2 \land C$
  - The check is done implicitly, as it is too expensive
    - No simplification, then all clauses are entailed
    - Only clause elimination / model increasing techniques, then sharing preserves equisatisfiability
Parallel Portfolio Solvers – Sharing

- Given two solvers $S_1$ and $S_2$, and let $S_1$ learn a clause $C$

- Let $F^1$ and $F^2$ be the working formulas of $S_1$ and $S_2$, respectively

- When is $S_2$ allowed to receive $C$
  - If $F^2 \equiv F^2 \land C$
  - If $F^2 \equiv_{SAT} F^2 \land C$
  - The check is done implicitly, as it is too expensive
    - No simplification, then all clauses are entailed
    - Only clause elimination / model increasing techniques, then sharing preserves equisatisfiability
  - Addition of redundant, but not entailed, clauses
    - Extra care, otherwise:
      - $F^1 = x$, and $S_2$ applies simplification:
        $F^2 = x \leadsto_{\text{modelInc}} F^2 = \emptyset \leadsto_{\text{modelDec}} F^2 = \overline{x}$
      - Solver $S_2$ receives $(x)$ from $S_1$: $F^2 = \overline{x} \land x$
      - Satisfiable formula is found to be unsatisfiable
Parallel Portfolio Solvers – Sharing

- Given two solvers $S_1$ and $S_2$, and let $S_1$ learn a clause $C$
- Let $F^1$ and $F^2$ be the working formulas of $S_1$ and $S_2$, respectively
- When is $S_2$ allowed to receive $C$
  - If $F^2 \equiv F^2 \land C$
  - If $F^2 \equiv_{\text{SAT}} F^2 \land C$
  - The check is done implicitly, as it is too expensive
    - No simplification, then all clauses are entailed
    - Only clause elimination / model increasing techniques, then sharing preserves equisatisfiability
  - Addition of redundant, but not entailed, clauses
    - Do not receive clauses, if a model decreasing has been used
Solving SAT in parallel with the Portfolio approach

- Different SAT solvers compete
Solving SAT in parallel with the Portfolio approach

The portfolio of these solvers requires the smallest run time
Solving SAT in parallel with the Portfolio approach

By adding communication among the solvers, the performance can be improved
High Level Parallelization Approaches

Parallel Portfolio Solvers
Search Space Partitioning Solvers
Search Space Partitioning

Partition search space of formula $F$ into sub spaces:

- For some $r > 0$ create $r$ “child”-formulas $F^i$, $0 < i \leq r$, such that
  - their disjunction is equal to the initial formula $F \equiv \bigvee F^i$
  - a partition constraint $K^i$ in CNF is added, $F^i := F \land K^i$
Search Space Partitioning

- Partition search space of formula $F$ into sub spaces:
  - For some $r > 0$ create $r$ “child”-formulas $F^i$, $0 < i \leq r$, such that
    - their disjunction is equal to the initial formula $F \equiv \bigvee F^i$
    - a partition constraint $K^i$ in CNF is added, $F^i := F \land K^i$
  - To obtain a partition, additionally ensure
    - that the child-formulas represent disjoint search spaces $F^i \land F^j \equiv \bot$, for all $0 \leq i < j \leq r$. 
Search Space Partitioning

- Partition search space of formula $F$ into sub spaces:
  - For some $r > 0$ create $r$ “child”-formulas $F_i$, $0 < i \leq r$, such that
    - their disjunction is equal to the initial formula $F \equiv \bigvee F_i$
    - a partition constraint $K^i$ in CNF is added, $F^i := F \land K^i$

- To obtain a partition, additionally ensure
  - that the child-formulas represent disjoint search spaces
    $F^i \land F^j \equiv \bot$, for all $0 \leq i < j \leq r.$

- Solve each child-formula with a sequential solver
- If a solver proofed unsatisfiability of a sub formula
  - assign a new child formula

Steffen Hölldobler and Norbert Manthey
SAT Solving – Parallel Search

INTERNATIONAL CENTER
FOR COMPUTATIONAL LOGIC
Search Space Partitioning

- Partition search space of formula $F$ into sub spaces:
  - For some $r > 0$ create $r$ “child”-formulas $F^i$, $0 < i \leq r$, such that
    - their disjunction is equal to the initial formula $F \equiv \bigvee F^i$
    - a partition constraint $K^i$ in CNF is added, $F^i := F \land K^i$
  - To obtain a partition, additionally ensure
    - that the child-formulas represent disjoint search spaces
      $F^i \land F^j \equiv \bot$, for all $0 \leq i < j \leq r$.
  - Solve each child-formula with a sequential solver
  - If a solver proofed unsatisfiability of a sub formula
    - assign a new child formula
  - Load-balancing is usually handled by providing sufficiently many child formulas
  - Can scale with the number of created child formulas, if partitioning works
(Plain) Search Space Partitioning

F

F\textsuperscript{1}

F\textsuperscript{2}

F\textsuperscript{3}

F\textsuperscript{4}

Solver(F\textsuperscript{1})

Solver(F\textsuperscript{2})

Solver(F\textsuperscript{3})

Solver(F\textsuperscript{4})

runtime

SAT

\perp

▶ finds models as fast as the fastest solver
(Plain) Search Space Partitioning

- proofs unsatisfiability as slow as the slowest solver
(Plain) Search Space Partitioning

\[ \text{not ensured: } \max(t_{\text{Solver}}(F^1), t_{\text{Solver}}(F^2), t_{\text{Solver}}(F^3), t_{\text{Solver}}(F^4)) \leq (t_{\text{Solver}}(F)) \]
Iterative Search Space Partitioning

- Partition search space of formula $F$ into sub spaces:
  - For some $r > 0$ create $r$ “child”-formulas $F^i$, $0 \leq i \leq r$, such that
    - $F \equiv \bigvee F^i$
    - $F^i \land F^j \equiv \perp$, for all $0 \leq i < j \leq r$.
  - Solve all formulas with a sequential solver (not only child formulas)
  - If a solver proofed unsatisfiability of a sub formula, assign a new child formula
    - assign a new child formula
    - or by recursively applying the partitioning procedure to child formulas
Iterative Search Space Partitioning

- Partition search space of formula $F$ into sub spaces:
  - For some $r > 0$ create $r$ “child”-formulas $F^i, 0 \leq i \leq r$, such that
    - $F \equiv \bigvee F^i$
    - $F^i \land F^j \equiv \bot$, for all $0 \leq i < j \leq r$.
  - Solve all formulas with a sequential solver (not only child formulas)
  - If a solver proofed unsatisfiability of a sub formula, assign a new child formula
    - assign a new child formula
    - or by recursively applying the partitioning procedure to child formulas
  - Creates a breadth first search in the search space
Iterative Search Space Partitioning

Partition search space of formula $F$ into sub spaces:

- For some $r > 0$ create $r$ “child”-formulas $F^i$, $0 \leq i \leq r$, such that
  - $F \equiv \bigvee F^i$
  - $F^i \land F^j \equiv \bot$, for all $0 \leq i < j \leq r$.

- Solve all formulas with a sequential solver (not only child formulas)
- If a solver proofed unsatisfiability of a sub formula, assign a new child formula
  - assign a new child formula
  - or by recursively applying the partitioning procedure to child formulas
- Creates a breadth first search in the search space
- Can assign new child-formulas to new resources by iterative partitioning
Iterative Search Space Partitioning

- $F_1$, $F_2$, $F_3$, $F_4$

- $\text{Solver}(F)$

- SAT

- runtime

- finds models as fast as the fastest solver
Iterative Search Space Partitioning

- \( F \)
- \( F^1 \)
- \( F^2 \)
- \( F^3 \)
- \( F^4 \)

Solver(\( F^1 \))
Solver(\( F^2 \))
Solver(\( F^3 \))
Solver(\( F^4 \))

runtime

\( ⊥ \)
\( UNSAT \)
\( ⊥ \)
\( ⊥ \)

- proofs unsatisfiability as fast as the slowest “necessary” solver
Iterative Search Space Partitioning

by iteratively partitioning the search space, new child formulas become more constrained
Iterative Partitioning – Dependencies

- Solve formula $F$
- Create a tree
  - Create partitioning constraints $K^i$ with $1 \leq i \leq r$, for some $r$
  - $F \equiv \bigvee_{1 \leq i \leq k} (F \land K^i)$
  - $K^i \land K^j \equiv \bot$ for all $1 \leq i < j \leq k$

- Definition
  - A clause $C$ depends on a path $p$, if $p$ is the longest path of all clauses that participated in the derivation of $C$. 
Iterative Partitioning – Dependencies

\[ F^p := ((x_1 \lor x_2 \lor x_5) \land (x_3 \lor x_4) \land (\overline{x_2}, x_6, x_1) \land (\overline{x_2} \lor \overline{x_6})) \]

\[ F^{p1} := ((x_2 \lor x_5) \land (x_3 \lor x_4) \land (\overline{x_2} \lor x_6) \land \ldots) \]

\[ F^{p2} := ((x_3 \lor x_4) \land \ldots) \]
Iterative Partitioning – Dependencies

\[
F^p := ((x_1 \lor x_2 \lor x_5) \land (x_3 \lor x_4) \land (\overline{x_2}, x_6, x_1) \land (\overline{x_2} \lor \overline{x_6}))
\]

\[
F^{p1} := ((x_2 \lor x_5) \land (x_3 \lor x_4) \land (\overline{x_2} \lor x_6) \land \ldots)
\]

\[
F^{p2} := ((x_3 \lor x_4) \land \ldots)
\]

The clauses \((x_2 \lor x_5)\) and \((\overline{x_2} \lor x_6)\) depend on the partitioning constraint.
Iterative Partitioning – Dependencies

\[ F^p := ((x_1 \lor x_2 \lor x_5) \land (x_3 \lor x_4) \land (\overline{x_2}, x_6, x_1) \land (\overline{x_2} \lor \overline{x_6})) \]

\[ F^{p1} := ((x_2 \lor x_5) \land (x_3 \lor x_4) \land (\overline{x_2} \lor x_6) \land \ldots) \]

\[ F^{p2} := ((x_3 \lor x_4) \land \ldots) \]

- The clauses \((x_2 \lor x_5)\) and \((\overline{x_2} \lor x_6)\) depend on the partitioning constraint.
- Their label has to be adapted accordingly!
Iterative Partitioning – A Abstract Example

- Solve formula $F$
- Create a tree
  - Create partitioning constraints $K^i$ with $1 \leq i \leq r$, for some $r$
  - $F \equiv \bigvee_{1 \leq i \leq k} (F \land K^i)$
  - $K^i \land K^j \equiv \bot$ for all $1 \leq i < j \leq k$
    - e.g. $K^1 = x \land y$, $K^2 = ((\neg x \lor \neg y) \land c)$ and $K^3 = ((\neg x \lor \neg y) \land \neg c)$
  - Label each node with its path to the root node,
    - e.g. $F^{132} = F \land K^1 \land K^{13} \land K^{132}$
Iterative Partitioning – A Abstract Example

- Solve formula $F$
- Create a tree
  - Create partitioning constraints $K^i$ with $1 \leq i \leq r$, for some $r$
  - $F \equiv \bigvee_{1 \leq i \leq k} (F \land K^i)$
  - $K^i \land K^j \equiv \bot$ for all $1 \leq i < j \leq k$
    - e.g. $K^1 = x \land y$, $K^2 = ((\neg x \lor \neg y) \land c)$ and $K^3 = ((\neg x \lor \neg y) \land \neg c)$
  - Label each node with its path to the root node,
    - e.g. $F^{132} = F \land K^1 \land K^{13} \land K^{132}$
- Have one solver for each core, assign a node to each solver
Iterative Partitioning – A Abstract Example

- Solve formula $F$
- Create a tree
  - Create partitioning constraints $K^i$ with $1 \leq i \leq r$, for some $r$
  - $F \equiv \bigvee_{1 \leq i \leq k} (F \land K^i)$
  - $K^i \land K^j \equiv \bot$ for all $1 \leq i < j \leq k$
    - e.g. $K^1 = x \land y$, $K^2 = ((\neg x \lor \neg y) \land c)$ and $K^3 = ((\neg x \lor \neg y) \land \neg c)$
  - Label each node with its path to the root node,
    - e.g. $F^{132} = F \land K^1 \land K^{13} \land K^{132}$
- Have one solver for each core, assign a node to each solver
- Partition nodes recursively if resources become available again
Iterative Partitioning – A Abstract Example

- Created 4 nodes with their partitioning constraints
- Assign all 5 solvers $S_1$ to $S_5$ to nodes
Iterative Partitioning – A Abstract Example

- Solver $S_2$ and $S_4$ find their formula to be unsatisfiable
- Partition $F^2$, and assign the solvers again
Iterative Partitioning – A Abstract Example

\[
\begin{align*}
F, S_1 & \equiv \bot \\
F^1 & \equiv \bot \\
F^2, S_3 & \equiv \bot \\
F^3 & \equiv \bot \\
F^4, S_5 & \equiv \bot \\n\end{align*}
\]
Iterative Partitioning – A Abstract Example

- Solver $S_2$ and $S_4$ find their formula to be unsatisfiable
- Assign $S_2$ to $F^{21}$, partition $F^4$, and assign $S_4$ to $F^{41}$
Iterative Partitioning – A Abstract Example

- Have one solver for each core, assign a node to each solver
- Partition nodes recursively if resources become available again

\[
\begin{align*}
F_1 &\equiv \bot \\
F_2, S_3 &
\end{align*}
\]

\[
\begin{align*}
F_1 &\equiv \bot \\
F_2 &\equiv \bot \\
F_3 &\equiv \bot \\
F_4, S_5 &
\end{align*}
\]

- Steffen Hölldobler and Norbert Manthey
- Steffen Hölldobler and Norbert Manthey

SAT Solving – Parallel Search
Iterative Partitioning – A Abstract Example

- Solver $S_2$ finds $F^{23}$ to be unsatisfiable
- $F^2$ has to be unsatisfiable as well
Iterative Partitioning – A Abstract Example

- Solver $S_2$ finds $F^{23}$ to be unsatisfiable
- $F^2$ has to be unsatisfiable as well
Iterative Partitioning – A Abstract Example

- Solver $S_4$ finds $F^{41}$ to be satisfiable
- Then $F^4$ and $F$ are satisfiable as well
Iterative Partitioning – A Abstract Example

- Solver $S_4$ finds $F^{41}$ to be satisfiable
- Then $F^4$ and $F$ are satisfiable as well
Iterative Partitioning – Downward Sharing

- Solver $S_1$ learns clause $C$
- Downward clause sharing is safe, $F \models C$, then $F \land K^i \models C$
- Assumption: no simplification

\[ F \models C \]
\[ F^4 = F \land K^4 \models C \]
\[ F^{41} = F \land K^4 \land K^{41} \models C \]
Iterative Partitioning – Upward Sharing

- Solver $S_2$ learns clause $C, F \land K^4 \land K^{42} \models C$
- Suppose $C$ depends only on $F$ and $K^4$
- Upward clause sharing to $F^4$ is safe
- Store dependency level for each clause
- Assumption: no simplification
Iterative Partitioning – Abort Redundant

- Solver $S_2$ learns empty clause $\perp$, $F \land K^4 \land K^{42} \models \perp$
- Suppose the empty clause depends only on $F$ and $K^4$
- Upward clause sharing to $F^4$ is safe
- Abort all solvers below $F^4$ ($S_2$, $S_4$ and $S_5$)

\[ F \models \perp \]
\[ F^4 = F \land K^4 \models \perp \]
\[ F^{41} = F \land K^4 \land K^{41} \models \perp \]
Partition Tree With Shared Clauses

\[ F = ((x_1 \lor x_2 \lor x_3) \land (\overline{x_3} \lor \overline{x_2})) \land (\overline{x_2} \lor x_4 \lor x_1) \land (\overline{x_2} \lor x_4 \lor \overline{x_1}) \land (x_4 \lor x_2 \lor x_5 \lor \overline{x_7}) \land (\overline{x_4} \lor x_2 \lor \overline{x_5}) \land (x_7 \lor x_8)) \]

\[ F^1 = (\ldots \land (\overline{x_2} \lor \overline{x_4} \lor \overline{x_1}) \land (x_4 \lor x_2 \lor x_5) \land (x_4 \lor x_2 \lor \overline{x_5})) \]

\[ F^2 = (\overline{x_7}) \]

\[ (x_7) \]

\[ (\overline{x_7}) \]

\[ (x_1) \]

\[ (\overline{x_1}) \]

\[ ((\overline{x_3} \lor \overline{x_2}) \land (\overline{x_2} \lor x_4) \land (x_4 \lor x_2 \lor \overline{x_4}) \ldots) \]

\[ ((x_2 \lor x_3) \land (\overline{x_3} \lor \overline{x_2}) \land \ldots) \]

\[ (x_3) \]

\[ (\overline{x_3}) \]

\[ F^{121} = ((\overline{x_2}) \land (x_4 \lor x_2 \lor x_5) \land (x_4 \lor x_2 \lor \overline{x_5})) \]

\[ ((x_2) \land (x_4 \lor x_2 \lor x_5) \land (x_4 \lor x_2 \lor \overline{x_5})) \]

\[ \blacktriangleright D = (x_4 \lor x_2) \text{ is learned by } (x_4 \lor x_2 \lor x_5) \otimes (x_4 \lor x_2 \lor \overline{x_5}) \text{ in formula } F^{121} \]

\[ \blacktriangleright D \text{ depends on the partition constraint } x_7 \]
Partition Tree With Shared Clauses

\[ F = \left( (x_1 \lor x_2 \lor x_3) \land (\overline{x_3} \lor \overline{x_2}) \right) \land \left( (\overline{x_2} \lor x_4 \lor \overline{x_1}) \land (\overline{x_2} \lor x_4 \lor \overline{x_1}) \right) \land \left( x_4 \lor x_2 \lor x_5 \lor \overline{x_7} \right) \land \left( x_4 \lor x_2 \lor \overline{x_5} \right) \land \left( x_7 \lor x_8 \right) \]

\[ F^1 = (\ldots \land \left( \overline{x_2} \lor \overline{x_4} \lor \overline{x_1} \right) \land \left( x_4 \lor x_2 \lor x_5 \right) \land \left( x_4 \lor x_2 \lor \overline{x_5} \right) \]

\[ F^2 \]

\[ F^{121} = \left( (\overline{x_2}) \land \left( x_4 \lor x_2 \lor x_5 \right) \right) \land \left( x_4 \lor x_2 \lor \overline{x_5} \right) \]

\[ (x_7) \quad (\overline{x_7}) \]

\[ (x_1) \quad (\overline{x_1}) \]

\[ ((\overline{x_3} \lor \overline{x_2}) \land (\overline{x_2} \lor x_4) \land (x_2 \lor x_4) \land \ldots) \]

\[ ((\overline{x_3} \lor \overline{x_2}) \land (\overline{x_2} \lor x_4) \land \ldots) \]

\[ (x_3) \quad (\overline{x_3}) \]

\[ D = (x_4 \lor x_2) \text{ is learned by } (x_4 \lor x_2 \lor x_5) \otimes (x_4 \lor x_2 \lor \overline{x_5}) \text{ in formula } F^{121} \]

\[ D \text{ depends on the partition constraint } x_7 \]

\[ \text{Hence, } D \text{ can be shared in the subtree of } F^1 \]
Not Discussed Here

- Sharing and Simplification
- Low-Level Parallelization
- Parallel Simplification