Example: DS-1 bus communication

- Some of the clauses describing bus communication

\[ C_1: \neg nci \lor \neg a \lor nco \quad C_4: \neg rf \lor ia \quad C_7: \neg ok \lor \neg uf \]

\[ C_2: \neg ia \lor nco \quad C_5: \neg uf \lor ia \quad C_8: \neg rf \lor \neg uf \]

\[ C_3: \neg ok \lor a \quad C_6: \neg ok \lor \neg rf \quad C_9: \neg a \lor \neg ia \]

BC health: \( ok, rf, uf \)  
BC activity: \( a, ia \)  
No input cmd: \( nci \)  
No output cmd: \( nco \)
LTMS inference

\[ C_7: \neg ok \lor \neg uf \]
\[ C_{10}: ok \]
\[ C_8: \neg rf \lor \neg uf \]
\[ C_9: \neg a \lor \neg ia \]
\[ C_3: \neg ok \lor a \]
\[ C_6: \neg ok \lor \neg rf \]
\[ C_4: \neg rf \lor ia \]
\[ C_2: \neg ia \lor nco \]

\[ C_1: \neg nci \lor \neg a \lor nco \]

uf \quad false

\[ \text{true} \quad nci \]

\[ \text{true} \quad \text{false} \quad \text{false} \quad \text{false} \]
Deleting $C_{10}$, adding $C_{11}$

$C_7: \neg ok \lor \neg uf$

$C_8: \neg rf \lor \neg uf$

$C_9: \neg a \lor \neg ia$

$C_3: \neg ok \lor a$

$C_6: \neg ok \lor \neg rf$

$C_4: \neg rf \lor ia$

$C_2: \neg ia \lor nco$

$C_1: \neg nci \lor \neg a \lor nco$

uf false false

ok true false

rf false true

true nco
Summary of problem

- **Problem:** Unnecessary repropagation during a context switch
- Excessive repropagation can be a significant problem in practice

Data from 387 distinct context switches on DS-1 theory containing 12,693 clauses

Ranges from 110% to 660% more label changes than necessary
Cause of problem

- To guarantee *well-founded support*, the LTMS context switch algorithm is overly conservative
  - switching a context by deleting clause $D$ and adding clause $A$

- **Previous solution:** Use an ATMS
  - context switch requires *no* label propagation
  - labeling algorithm is exponential in time and space

- **New solution:** Use an ITMS
ITMS algorithm intuitions

• *Resupport* propositions during clause deletion
  – if resupport is possible, proposition’s consequences are not touched
  – must guarantee *well-founded support*
• …but resupport available only after clause addition
  – so add new clause and propagate *before* deleting old clause
• …but added clause is often a conflict and propagation terminates
  – develop a new algorithm to *propagate through conflicts*
Top-level ITMS algorithm

procedure switch-context(D, A, Σ)    // delete D, add A to Σ
    Add A to Σ and propagate any unit clauses
    if conflicting clause detected then
        while there is a conflict that can be propagated do
            Propagate through the conflict
            Use propagated label to resupport propositions (if possible)
        endwhile
    endif
    Delete D from Σ
    Propagate any unit clauses
end switch-context
Resupporting a proposition

- Clause $R$ can resupport proposition $p$ which is currently supported by clause $C$ if
  - $p$ occurs with the same sign in $R$ and $C$
  - all other literals in $R$ are $false$
  - resulting support is $well-founded$

- Guaranteeing well-founded support is linear in the size of $\Sigma$
  - defeats the very purpose of using an ITMS
Propagation numbers

- Assign a *propagation number* to each supported proposition
  - proposition’s propagation number is *greater than* propagation number of other propositions occurring in supporting clause

  \[ \text{Propagation number increases monotonically} \]

\[
\begin{align*}
  C_1: r & \rightarrow r \quad \text{true: 1} \\
  C_2: \neg r \lor q & \rightarrow q \quad \text{true: 2} \\
  C_3: \neg q \lor \neg p & \rightarrow p \quad \text{false: 3}
\end{align*}
\]

⇒ If p’s propagation number is greater than q’s propagation number, then q cannot depend on p

- Resupport proposition p with clause R only if p’s propagation number is *greater than* propagation number of other literals in R
  - sufficient, but not necessary, condition for resupport
Resupporting \( nco \)

\[
\begin{align*}
C_7: & \, \neg ok \lor \neg uf \\
C_{10}: & \, ok \\
C_8: & \, \neg rf \lor \neg uf \\
C_{11}: & \, rf \\
C_1: & \, \neg nc_i \lor \neg a \lor nco \\
C_4: & \, \neg rf \lor ia \\
C_2: & \, \neg ia \lor nco \\
C_3: & \, \neg ok \lor a \\
C_6: & \, \neg ok \lor \neg rf \\
C_9: & \, \neg a \lor \neg ia \\
\end{align*}
\]
Propagating through a conflict

- Switch the label of a proposition $p$ in a conflict $C$
  - and let $p$’s support be $C$

- $C$ must provide $p$ with a well-founded support
  - $p$’s propagation number must be greater than or equal to the propagation number of other literals in $C$

- Resupport other propositions using clauses in which $p$ occurs
- Prevent infinite loops by changing a proposition’s label at most once
Propagating through $C_4$

$C_7: \neg ok \lor \neg uf$

$C_{10}: ok$

$C_8: \neg rf \lor \neg uf$

$C_{11}: rf$

$C_3: \neg ok \lor a$

$C_6: \neg ok \lor \neg rf$

$C_4: \neg rf \lor ia$

$C_9: \neg a \lor \neg ia$

$C_1: \neg nci \lor \neg a \lor nco$

$C_i: \neg nci \lor \neg a \lor nco$

uf false:2

ok true:1

rf false:2

true:1

ia false:3

true:5 nco
ITMS significantly decreases extra label changes

Data from 387 context switches on DS-1 theory containing 12,693 clauses

Ranges from 110% to 660% more label changes than necessary
Comparing the ITMS to the LTMS

![Bar chart showing the ratio of ITMS label changes to LTMS label changes (in percent). The x-axis represents the ratio, ranging from 10 to 110, with the y-axis representing the number of occurrences. The chart highlights a significant peak at around 70% ratio.]
Conclusions

• The ITMS is an aggressive incremental TMS that optimizes context switching
  – clause addition done before clause deletion
  – novel resupport algorithm using propagation numbers
  – novel algorithm to propagate through conflicts
• Dramatic reduction in worst-case performance compared to a traditional LTMS
• Critical for achieving adequate performance in Livingstone’s real-time propositional reasoning execution kernel