Decision Procedures for Concurrent Skiplists

Alejandro Sánchez\textsuperscript{1} \hspace{1cm} César Sánchez\textsuperscript{1,2}

\textsuperscript{1}The IMDEA Software Institute, Spain
\textsuperscript{2}Spanish Council for Scientific Research (CSIC), Spain

EPFL, Lausanne, 16 September 2010
Why do we want a decision procedure

- Imperative programs
Why do we want a decision procedure

- Imperative programs
- Concurrent data-structures

\[ P_1 || P_2 || \cdots || P_n \]
Why do we want a decision procedure

- Imperative programs
- Concurrent data-structures

\[
P_1 || P_2 || \cdots || P_n
\]

\[
\begin{array}{c}
\text{data structures} \\
(\text{heap})
\end{array}
\]
Why do we want a decision procedure

- Imperative programs
- Concurrent data-structures
- Temporal properties (safety, liveness)

\[
P_1 \parallel P_2 \parallel \cdots \parallel P_n \models \varphi
\]

\[\text{data structures (heap)}\]
Why do we want a decision procedure

- Imperative programs
- Concurrent data-structures
- Temporal properties (safety, liveness)
- Formal verification

\[ P_1 \parallel P_2 \parallel \cdots \parallel P_n \models \varphi \]

Regional Logic \rightarrow data structures (heap) \rightarrow LTL (□, ◇, U, \ldots)
Why do we want a decision procedure

- Imperative programs
- Concurrent data-structures
- Temporal properties (safety, liveness)
- Formal verification

\[
P_1 \parallel P_2 \parallel \cdots \parallel P_n \models \varphi
\]

Regional Logic \rightarrow data structures (heap) \rightarrow LTL (\Box, \Diamond, U, \ldots)
Verification of Concurrent Data-structures

Main Idea

Concurrent DataStructure
Verification of Concurrent Data-structures

Main Idea

Concurrent DataStructure

Most General Client
Verification of Concurrent Data-structures

Main Idea

Concurrent DataStructure

Most General Client

\[ P[N] : P(1) \parallel \cdots \parallel P(N) \]
Verification of Concurrent Data-structures

Main Idea

Concurrent DataStructure

Most General Client

\[ P[N] : P(1) \parallel \cdots \parallel P(N) \]

+ ghost variables
Verification of Concurrent Data-structures

Main Idea

Concurrent DataStructure

Most General Client

Property

$\varphi^{(k)}$

$P[N] : P(1) \parallel \cdots \parallel P(N)$

+ ghost variables
Verification of Concurrent Data-structures

Main Idea

Concurrent DataStructure

Most General Client

Diagram

Property

$P[N] : P(1) \parallel \cdots \parallel P(N)$

$\varphi^{(k)}$

ghost variables
Verification of Concurrent Data-structures

Main Idea

Concurrent DataStructure

Most General Client

\[ P[N] : P(1) \parallel \cdots \parallel P(N) \]

\[ + \quad \text{ghost variables} \]

Diagram

\[ \models \]

\[ D \]

Property

\[ \models \]

\[ \varphi^{(k)} \]

Verification Conditions:
- Initiation
- Consecution
- Acceptance
- Fairness

Satisfaction (Model Checking)
Verification of Concurrent Data-structures

Main Idea

Concurrent DataStructure

Most General Client

\[ P[N] : \langle P(1) \parallel \cdots \parallel P(N) \rangle \]

\[ + \]

ghost variables

Verification Conditions:

- Initiation
- Consecution
- Acceptance
- Fairness

Decision Procedures

Diagram

\[ \models \]

\( \mathcal{D} \)

\[ \models \]

\( \varphi^{(k)} \)

Satisfaction (Model Checking)
Verification Conditions

- *Initiation*
  \[ \Theta \rightarrow \mu(N_0) \]
Verification Conditions

- **Initiation**
  \[ \Theta \rightarrow \mu(N_0) \]

- **Consecution**: for all \( n \) and \( \tau \):
  \[ \mu(n)(s) \land \rho_\tau(s, s') \rightarrow \mu(\text{next}(n))(s') \]
Verification Conditions

▶ **Initiation**
\[ \Theta \rightarrow \mu(N_0) \]

▶ **Consecution:** for all \( n \) and \( \tau \):
\[ \mu(n)(s) \land \rho_\tau(s, s') \rightarrow \mu(\text{next}(n))(s') \]

▶ **Acceptance:** if \((n_1, n_2) \in P \setminus R\) then
\[ \mu(n_1)(s) \land \mu(n_2)(s') \land \rho_\tau(s, s') \rightarrow \delta_{n_1}(s) \geq \delta_{n_2}(s') \]
and if \((n_1, n_2) \notin P \cup R\):
\[ \mu(n_1)(s) \land \mu(n_2)(s') \land \rho_\tau(s, s') \rightarrow \delta_{n_1}(s) > \delta_{n_2}(s') \]
Verification Conditions

- **Initiation**
  \[ \Theta \rightarrow \mu(N_0) \]

- **Consecution**: for all \( n \) and \( \tau \):
  \[ \mu(n)(s) \land \rho_{\tau}(s,s') \rightarrow \mu(\text{next}(n))(s') \]

- **Acceptance**: if \((n_1, n_2) \in P \setminus R\) then
  \[ \mu(n_1)(s) \land \mu(n_2)(s') \land \rho_{\tau}(s,s') \rightarrow \delta_{n_1}(s) \geq \delta_{n_2}(s') \]
  and if \((n_1, n_2) \notin P \cup R\):
  \[ \mu(n_1)(s) \land \mu(n_2)(s') \land \rho_{\tau}(s,s') \rightarrow \delta_{n_1}(s) > \delta_{n_2}(s') \]

- **Fairness**: for all \( n \) and \( \tau \in \eta(n,n') \):
  \[ \mu(n)(s) \rightarrow En_{\tau}(s) \]
  \[ \mu(n)(s) \land \rho_{\tau}(s,s') \rightarrow \mu(\tau(n))(s') \]
Concurrent Lock-Coupling SkipLists
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists

![Diagram of skiplist with elements: -∞, 5, 8, 12, 22, 53, 70, +∞]
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)

\( \text{insert}(60) \) with height 1
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)

\[
\text{insert}(60) \text{ with height } 1
\]
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)

$\text{insert}(60)$ with height 1
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)

\[ \text{insert}(60) \text{ with height 1} \]
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)

\[\text{insert}(60)\text{ with height 1}\]
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)

\[\text{insert}(60)\text{ with height 1}\]
Concurrent Lock-Coupling Skip Lists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)

\[\text{insert}(60)\text{ with height 1}\]
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)

\[\text{insert}(60)\text{ with height 1}\]
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)

\[ \text{insert}(60) \text{ with height 1} \]
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)

\textit{insert}(60) with height 1
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)

$\text{insert}(60)$ with height 1
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)

\[\text{insert}(60) \text{ with height 1}\]
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)

\[\text{insert}(60) \text{ with height 1}\]

```
\begin{array}{cccccc}
& & & & & \\
& & & & & \\
& & & & & \\
& & & & & \\
\text{head} & 5 & 8 & 12 & 22 & 53 & 70 & \text{tail} \\
\end{array}
```
Concurrent Lock-Coupling SkipLists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)

\[ \text{insert}(60) \text{ with height 1} \]
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)

\[\text{insert}(60)\text{ with height 1}\]
Verification Conditions Examples
Verification Conditions Examples

- Preservation of skiplistness shape
Verification Conditions Examples

- Preservation of skiplistness shape

\[
\text{SkipList}_3(sl : \text{SkipList}) \equiv \\
\text{OList}_0(h, sl.\text{head}, sl.r_0) \land \\
\text{OList}_1(h, sl.\text{head}, sl.r_1) \land \\
\text{OList}_2(h, sl.\text{head}, sl.r_2) \land \\
\pi_1(sl.r_3) \subseteq \pi_1(sl.r_2) \subseteq \pi_1(sl.r_1) \land \\
sl.\text{last}.\text{next}_0 = \text{null} \land sl.\text{last}.\text{next}_1 = \text{null} \land \\
sl.\text{last}.\text{next}_2 = \text{null} \land \\
\text{SubPath}(\text{getp}_1(h, sl.\text{head}, sl.\text{last}), \text{getp}_0(h, sl.\text{head}, sl.\text{last})) \land \\
\text{SubPath}(\text{getp}_2(h, sl.\text{head}, sl.\text{last}), \text{getp}_1(h, sl.\text{head}, sl.\text{last}))
\]
Verification Conditions Examples

- Preservation of skiplistness shape
- Program transitions
Verification Conditions Examples

- Preservation of skiplistness shape
- Program transitions

\[
\text{SkipList}_3(sl) \land \text{at}\_\text{insert}_{31} \land 0 \leq i \leq 2 \land \\
x.\text{val} = v \land \text{update}[i].\text{val} < v \land \\
\text{update}[i].\text{next}[i].\text{val} > v \land x.\text{next}[i] = \text{update}[i].\text{next}[i] \land \\
m_r = \{(\text{update}[i], i), (x.\text{next}[i], i)\} \cup m_{i+1..2} \land \text{update}[i].\text{locks}[i] = t \land \\
\text{update}[i].\text{next}[i].\text{locks}[i] = t \land (j < i \rightarrow (x, i) \in sl.r_j) \land \\
\text{update}'[i].\text{next}[i] := x \land sl'.r_i := sl.r_i \cup \{(x, i)\} \rightarrow \\
\text{SkipList}_3(sl') \land \text{at}'\_\text{insert}_{32} \land \text{update}'[i].\text{key} < k \land \\
\text{update}'[i].\text{next}[i].\text{next}[i].\text{key} > k \land \\
x'.\text{next}[i] = \text{update}'[i].\text{next}[i].\text{next}[i] \land \\
\text{update}'[i].\text{next}[i] = x' \land \\
m'_r = \{(\text{update}'[i], i), (x'.\text{next}[i], i)\} \cup m'_{i+1..2} \land \\
\text{update}'[i].\text{locks}[i] = t \land \text{update}'[i].\text{next}[i].\text{next}[i].\text{locks}[i] = t
\]
Theory of Concurrent Skiplists of Height $K$ (TSL$_K$)
Theory of Concurrent Skiplists of Height $K$ ($\text{TSL}_K$)

- Based on TLL
Theory of Concurrent Skiplists of Height $K$ ($TSL_K$)

- Based on TLL
- Extend all possible reasoning up to $K$ levels
Theory of Concurrent Skiplists of Height K ($TSL_K$)

- Based on TLL
- Extend all possible reasoning up to K levels
- Add the possibility of working with masked regions
Theory of Concurrent Skiplists of Height $K$ (TSL$_K$)

- Based on TLL
- Extend all possible reasoning up to $K$ levels
- Add the possibility of working with masked regions
- Description of order in lists and sub-paths
Theory of Concurrent Skiplists of Height $K$ ($TSL_K$)
Theory of Concurrent Skiplists of Height $K$ ($TSL_K$)

- Union of theories
Theory of Concurrent Skiplists of Height $K$ ($\text{TSL}_K$)

- Union of theories

\[
\text{TSL}_K = T_{\text{addr}}
\]

\[
\Sigma_{\text{addr}} = \{\text{addr}\}, \emptyset, \emptyset
\]
Theory of Concurrent Skiplists of Height $K$ ($TSL_K$)

- Union of theories

\[
TSL_K = T_{\text{addr}} \oplus T_{\text{elem}}
\]

\[\Sigma_{\text{elem}} = \{\text{elem}\}, \emptyset, \emptyset\]
Theory of Concurrent Skiplists of Height $K$ (TSL$_K$)

- Union of theories

\[ TSL_K = T_{addr} \oplus T_{elem} \oplus T_{cell} \]

\[ \Sigma_{cell} = \{ \text{cell, elem, ord, addr, thid} \} \]

- `error` : cell
- `mkcell` : elem $\times$ ord $\times$ addr$^K$ $\times$ thid$^K$ $\rightarrow$ cell
- `_.data` : cell $\rightarrow$ elem
- `_.key` : cell $\rightarrow$ ord
- `_.next[.]` : cell $\times$ level$^K$ $\rightarrow$ addr
- `_.lockid[.]` : cell $\times$ level$^K$ $\rightarrow$ thid
- `_.lock[.]` : cell $\times$ level$^K$ $\rightarrow$ thid $\rightarrow$ cell
- `_.unlock[.]` : cell $\times$ level$^K$ $\rightarrow$ cell
Theory of Concurrent Skiplists of Height $K$ ($\text{TSL}_K$)

- Union of theories

\[
\text{TSL}_K = T_{\text{addr}} \oplus T_{\text{elem}} \oplus T_{\text{cell}} \oplus T_{\text{mem}}
\]

\[
\Sigma_{\text{mem}} = \{\text{mem, addr, cell}\} \\
\begin{align*}
\text{null} : & \quad \text{addr} \\
\text{-}[\cdot] : & \quad \text{mem} \times \text{addr} \to \text{cell} \\
\text{upd} : & \quad \text{mem} \times \text{addr} \times \text{cell} \to \text{mem}
\end{align*}
\]

\[
\emptyset
\]
Theory of Concurrent Skiplists of Height $K$ (TSL$_K$)

- Union of theories

$$TSL_K = T_{addr} \oplus T_{elem} \oplus T_{cell} \oplus T_{mem} \oplus T_{Reachability}$$

$$\Sigma_{Reachability} = \begin{cases} 
\{ \text{mem, addr, path} \} \\
\epsilon & : \text{path} \\
[-] & : \text{addr} \rightarrow \text{path} \\
append & : \text{path} \times \text{path} \times \text{path} \\
\text{reach}_K & : \text{mem} \times \text{addr} \times \text{addr} \times \text{level}_K \times \text{path} 
\end{cases}$$
Theory of Concurrent Skiplists of Height $K$ ($TSL_K$)

- Union of theories

\[
TSL_K = \sum_{\text{addr}} \oplus T_{\text{elem}} \oplus T_{\text{cell}} \oplus T_{\text{mem}} \oplus T_{\text{Reachability}} \oplus T_{\text{set}}
\]

\[
\Sigma_{\text{set}} = \begin{cases}
\{ \text{addr, set} \} \\
\emptyset & : \text{set} \\
\{-\} & : \text{addr} \to \text{set} \\
\cup, \cap, \setminus & : \text{set} \times \text{set} \to \text{set} \\
\in & : \text{addr} \times \text{set} \\
\subseteq & : \text{set} \times \text{set}
\end{cases}
\]
Theory of Concurrent Skiplists of Height $K$ ($TSL_K$)

- Union of theories

$$TSL_K = T_{addr} \oplus T_{elem} \oplus T_{cell} \oplus T_{mem} \oplus T_{Reachability} \oplus T_{set} \oplus T_{setth}$$

$$\Sigma_{setth} = \{\text{thid, setth}\}$$

$$\emptyset_T : \text{setth}$$

$$\{-\}_T : \text{thid} \rightarrow \text{setth}$$

$$\cup_T, \cap_T, \setminus_T : \text{setth} \times \text{setth} \rightarrow \text{setth}$$

$$\in_T : \text{thid} \times \text{setth}$$

$$\subseteq_T : \text{setth} \times \text{setth}$$
Theory of Concurrent SkipLists of Height $K$ ($TSL_K$)

- Union of theories

$$TSL_K = T_{addr} \oplus T_{elem} \oplus T_{cell} \oplus T_{mem} \oplus T_{Reachability} \oplus T_{set} \oplus T_{setth} \oplus T_{thid}$$

$$\Sigma_{setth} = \{ \text{thid} \}, \emptyset, \emptyset$$
The Theory of Concurrent Skiplists of Height $K$ ($\text{TSL}_K$)

- Union of theories

$$\text{TSL}_K = T_{\text{addr}} \oplus T_{\text{elem}} \oplus T_{\text{cell}} \oplus T_{\text{mem}} \oplus T_{\text{Reachability}} \oplus T_{\text{set}} \oplus T_{\text{setth}} \oplus T_{\text{thid}} \oplus T_{\text{mrgn}}$$

$$\Sigma_{\text{mrgn}} = \left\{ \text{mrgn, addr, level}_K \right\}$$

$$\begin{align*}
\text{emp}_{\text{mr}} & : \text{mrgn} \\
\langle -, - \rangle_{\text{mr}} & : \text{addr} \times \text{level}_K \to \text{mrgn} \\
\cup_{\text{mr}}, \cap_{\text{mr}}, -_{\text{mr}} & : \text{mrgn} \times \text{mrgn} \to \text{mrgn} \\
\in_{\text{mr}} & : \text{addr} \times \text{level}_K \times \text{mrgn} \\
\subseteq_{\text{mr}} & : \text{mrgn} \times \text{mrgn} \\
\#_{\text{mr}} & : \text{mrgn} \times \text{mrgn}
\end{align*}$$
Theory of Concurrent Skiplists of Height $K$ ($TSL_K$)

- Union of theories

$$TSL_K = T_{addr} \oplus T_{elem} \oplus T_{cell} \oplus T_{mem} \oplus T_{Reachability} \oplus T_{set} \oplus T_{setth} \oplus T_{thid} \oplus T_{mrgn} \oplus T_{ord}$$

$$\Sigma_{ord} = \begin{cases} \{\text{ord}\} \\ 0 : \text{ord} \\ +\infty : \text{ord} \\ \preceq : \text{ord} \times \text{ord} \end{cases}$$
Theory of Concurrent Skiplists of Height \( K \) (TSL\(_K\))

- Union of theories

\[
\text{TSL}_K = \text{T}_{\text{addr}} \oplus \text{T}_{\text{elem}} \oplus \text{T}_{\text{cell}} \oplus \text{T}_{\text{mem}} \oplus \text{T}_{\text{Reachability}} \oplus \\
\text{T}_{\text{set}} \oplus \text{T}_{\text{setth}} \oplus \text{T}_{\text{thid}} \oplus \text{T}_{\text{mrgn}} \oplus \text{T}_{\text{ord}} \oplus \text{T}_{\text{level}_K}
\]

\[
\Sigma_{\text{level}_K} = \{ \text{level}_K \} \\
\quad \begin{cases} \\
\quad \quad \eta_1 : \text{level} \\
\quad \quad \vdots \\
\quad \quad \eta_K : \text{level} \\
\quad \emptyset \\
\end{cases}
\]
Theory of Concurrent Skiplists of Height $K$ ($\text{TSL}_K$)

- Union of theories

$$\text{TSL}_K = T_{\text{addr}} \oplus T_{\text{elem}} \oplus T_{\text{cell}} \oplus T_{\text{mem}} \oplus T_{\text{Reachability}} \oplus T_{\text{set}} \oplus T_{\text{setth}} \oplus T_{\text{thid}} \oplus T_{\text{mrgn}} \oplus T_{\text{ord}} \oplus T_{\text{level}_K}$$

$$\Sigma_{\text{Bridge}} = \{ \text{mem, addr, set, path} \}$$

$$\begin{aligned}
\text{path2set} & : \text{path} \rightarrow \text{set} \\
\text{addr2set}_K & : \text{mem} \times \text{addr} \times \text{level}_K \rightarrow \text{set} \\
\text{getp}_K & : \text{mem} \times \text{addr} \times \text{addr} \times \text{level}_K \rightarrow \text{path} \\
\text{firstlocked}_K & : \text{mem} \times \text{path} \times \text{level}_K \rightarrow \text{addr} \\
\text{ordList} & : \text{mem} \times \text{path} \\
\end{aligned}$$
Theory of Concurrent Skiplists of Height $K$ ($\text{TSL}_K$)

- Union of theories

\[
\text{TSL}_K = \ T_{\text{addr}} \oplus T_{\text{elem}} \oplus T_{\text{cell}} \oplus T_{\text{mem}} \oplus T_{\text{Reachability}} \oplus T_{\text{set}} \oplus T_{\text{setth}} \oplus T_{\text{thid}} \oplus T_{\text{mrgn}} \oplus T_{\text{ord}} \oplus T_{\text{level}_K}
\]
Theory of Concurrent Skiplists of Height $K$ ($TSL_K$)

- Union of theories

$$TSL_K = T_{addr} \oplus T_{elem} \oplus T_{cell} \oplus T_{mem} \oplus T_{Reachability} \oplus T_{set} \oplus T_{setth} \oplus T_{thid} \oplus T_{mrgn} \oplus T_{ord} \oplus T_{level_K}$$

- We want to use Nelson-Oppen
Theory of Concurrent Skiplists of Height $K$ ($TSL_K$)

- Union of theories

$$TSL_K = T_{addr} \oplus T_{elem} \oplus T_{cell} \oplus T_{mem} \oplus T_{Reachability} \oplus$$
$$T_{set} \oplus T_{setth} \oplus T_{thid} \oplus T_{mrgn} \oplus T_{ord} \oplus T_{level_K}$$

- We want to use Nelson-Oppen

  - Stable infinite
Theory of Concurrent Skiplists of Height $K$ ($TSL_K$)

- Union of theories

$$TSL_K = T_{addr} \oplus T_{elem} \oplus T_{cell} \oplus T_{mem} \oplus T_{Reachability} \oplus T_{set} \oplus T_{setth} \oplus T_{thid} \oplus T_{mrgn} \oplus T_{ord} \oplus T_{level_K}$$

- We want to use Nelson-Oppen
  - Stable infinite
  - Disjoint signature (except by sort)
Theory of Concurrent Skiplists of Height $K$ (TSL$_K$)

- Union of theories

\[
\text{TSL}_K = T_{\text{addr}} \oplus T_{\text{elem}} \oplus T_{\text{cell}} \oplus T_{\text{mem}} \oplus T_{\text{Reachability}} \oplus T_{\text{set}} \oplus T_{\text{setth}} \oplus T_{\text{thid}} \oplus T_{\text{mrgn}} \oplus T_{\text{ord}} \oplus T_{\text{level}_K}
\]

- We want to use Nelson-Oppen
  - Stable infinite
  - Disjoint signature (except by sort)
  - Decision procedure
Theory of Concurrent Skiplists of Height $K$ ($\text{TSL}_K$)

- Union of theories

$$\text{TSL}_K \ = \ T_{\text{addr}} \oplus T_{\text{elem}} \oplus T_{\text{cell}} \oplus T_{\text{mem}} \oplus T_{\text{Reachability}} \oplus T_{\text{set}} \oplus T_{\text{setth}} \oplus T_{\text{thid}} \oplus T_{\text{mrgn}} \oplus T_{\text{ord}} \oplus T_{\text{level}}_K$$

- We want to use Nelson-Oppen
  - Stable infinite
  - Disjoint signature (except by sort)
  - Decision procedure

- Problem
Theory of Concurrent Skiplists of Height K (TSL\(_K\))

- Union of theories

\[
\text{TSL}_K = T_{\text{addr}} \oplus T_{\text{elem}} \oplus T_{\text{cell}} \oplus T_{\text{mem}} \oplus T_{\text{Reachability}} \oplus T_{\text{set}} \oplus T_{\text{sett}} \oplus T_{\text{thid}} \oplus T_{\text{mrgn}} \oplus T_{\text{ord}} \oplus T_{\text{level}_K}
\]

- We want to use Nelson-Oppen
  - Stable infinite
  - Disjoint signature (except by sort)
  - Decision procedure

- Problem
  - No decision procedure for reachability
Theory of Concurrent Skiplists of Height $K$ ($TSL_K$)

- Union of theories

$$TSL_K = T_{addr} \oplus T_{elem} \oplus T_{cell} \oplus T_{mem} \oplus T_{Reachability} \oplus T_{set} \oplus T_{setth} \oplus T_{thid} \oplus T_{mrgn} \oplus T_{ord} \oplus T_{level_K}$$

- We want to use Nelson-Oppen
  - Stable infinite
  - Disjoint signature (except by sort)
  - Decision procedure

- Problem
  - No decision procedure for reachability
  - Eliminate bridge functions and predicates, preserving satisfiability
Small Model Property
Small Model Property

Given a theory $T$ with $\Sigma = (S, F, P)$ and $S_0 \subseteq S$

$T$ has SMP with respect to $S_0$, if for every $T$-satisfiable QF $\Sigma$-formula $\varphi$ exists $T$-interpretation $A$ satisfying $\varphi$ s.t. $A_\sigma$ is finite, for every $\sigma \in S_0$
Small Model Property

Given a theory $T$ with $\Sigma = (S, F, P)$ and $S_0 \subseteq S$

$T$ has SMP with respect to $S_0$, if for every $T$-satisfiable QF $\Sigma$-formula $\varphi$ exists $T$-interpretation $A$ satisfying $\varphi$ s.t. $A_\sigma$ is finite, for every $\sigma \in S_0$

$\Gamma$ a conjunction of TSL$_K$-literals
Small Model Property

Given a theory $T$ with $\Sigma = (S, F, P)$ and $S_0 \subseteq S$,

$T$ has SMP with respect to $S_0$, if for every $T$-satisfiable QF $\Sigma$-formula $\varphi$ exists $T$-interpretation $A$ satisfying $\varphi$ s.t. $A_\sigma$ is finite, for every $\sigma \in S_0$

$$\Gamma \quad \text{a conjunction of TSL}_K\text{-literals}$$

$$\downarrow$$

$$\text{a conjunction of normalized TSL}_K\text{-literals}$$
Small Model Property

Given a theory $T$ with $\Sigma = (S, F, P)$ and $S_0 \subseteq S$

$T$ has SMP with respect to $S_0$, if for every $T$-satisfiable QF $\Sigma$-formula $\varphi$ exists $T$-interpretation $A$ satisfying $\varphi$ s.t. $A_\sigma$ is finite, for every $\sigma \in S_0$

\[
\Gamma \quad \text{a conjunction of TSL}_K\text{-literals}
\]
\[
\downarrow
\]
\[
a \text{ a conjunction of normalized TSL}_K\text{-literals}
\]

- Proof that exists a TSL$_K$-interpretation $A$
  - Bounded on $K$ and $\Gamma$.
  - With finite number of elements in addr, elem, thid, ord and level$_K$. 
Small Model Property

Given a theory $T$ with $\Sigma = (S, F, P)$ and $S_0 \subseteq S$.

$T$ has SMP with respect to $S_0$, if for every $T$-satisfiable QF $\Sigma$-formula $\phi$ exists $T$-interpretation $A$ satisfying $\phi$ s.t. $A_\sigma$ is finite, for every $\sigma \in S_0$.

$\Gamma$ a conjunction of TSL$_K$-literals

\\[ \downarrow \]

a conjunction of normalized TSL$_K$-literals

- Proof that exists a TSL$_K$-interpretation $A$
  - Bounded on $K$ and $\Gamma$.
  - With finite number of elements in addr, elem, thid, ord and level$_K$.

$\Gamma$ is also $T$-satisfiable in $A$.
Small Model Property

Given a theory $T$ with $\Sigma = (S, F, P)$ and $S_0 \subseteq S$

$T$ has SMP with respect to $S_0$, if for every $T$-satisfiable QF $\Sigma$-formula $\varphi$ exists $T$-interpretation $A$ satisfying $\varphi$ s.t. $A_\sigma$ is finite, for every $\sigma \in S_0$

$\Gamma$ a conjunction of $\text{TSL}_K$-literals

\[ \downarrow \]

a conjunction of normalized $\text{TSL}_K$-literals

- Proof that exists a $\text{TSL}_K$-interpretation $A$
  - Bounded on $K$ and $\Gamma$.
  - With finite number of elements in $\text{addr}$, $\text{elem}$, $\text{thid}$, $\text{ord}$ and $\text{level}_K$.

$\Gamma$ is also $T$-satisfiable in $A$

- $\text{TSL}_K$ enjoys the small model property
A Decision Procedure for $\text{TSL}_K$
A Decision Procedure for $\text{TSL}_K$

$$\text{TSL}_K = \cdots \oplus T_{\text{Reachability}} \oplus \cdots \oplus \text{bridge functions and predicates}$$
A Decision Procedure for $\text{TSL}_K$

$\text{TSL}_K = \cdots \oplus T\text{Reachability} \oplus \cdots \oplus \text{bridge functions and predicates}$

$T_{\text{Base}} = T_{\text{addr}} \oplus T_{\text{elem}} \oplus T_{\text{cell}} \oplus T_{\text{mem}} \oplus T_{\text{fseq}} \oplus$

$T_{\text{set}} \oplus T_{\text{setth}} \oplus T_{\text{thid}} \oplus T_{\text{mrgn}} \oplus T_{\text{ord}} \oplus T_{\text{level}_K}$
A Decision Procedure for TSL$_K$

\[ TSL_K = \cdots \oplus T_{\text{Reachability}} \oplus \cdots \oplus \text{bridge functions and predicates} \]

\[ T_{\text{Base}} = T_{\text{addr}} \oplus T_{\text{elem}} \oplus T_{\text{cell}} \oplus T_{\text{mem}} \oplus T_{\text{fseq}} \oplus T_{\text{path}} \]

\[ T_{\text{set}} \oplus T_{\text{setth}} \oplus T_{\text{thid}} \oplus T_{\text{mrgn}} \oplus T_{\text{ord}} \oplus T_{\text{level}_K} \]
A Decision Procedure for $\text{TSL}_K$

$\text{TSL}_K = \cdots \oplus T_{\text{Reachability}} \oplus \cdots \oplus \text{bridge functions and predicates}$

$T_{\text{Base}} = T_{\text{addr}} \oplus T_{\text{elem}} \oplus T_{\text{cell}} \oplus T_{\text{mem}} \oplus T_{\text{fseq}} \oplus T_{\text{set}} \oplus T_{\text{setth}} \oplus T_{\text{thid}} \oplus T_{\text{mrgn}} \oplus T_{\text{ord}} \oplus T_{\text{level}_K}$
A Decision Procedure for $\text{TSL}_K$

$\text{TSL}_K = \cdots \oplus T_{\text{Reachability}} \oplus \cdots \oplus \text{bridge functions and predicates}$

\[
\begin{align*}
\text{TSL}_K & = T_{\text{Base}} \oplus T_{\text{set}} \oplus T_{\text{setth}} \oplus T_{\text{thid}} \oplus T_{\text{mrgn}} \oplus T_{\text{ord}} \oplus T_{\text{level}_{K}} \oplus T_{\text{fseq}} \oplus T_{\text{path}} \\
& \leq \text{normalized TSL}_K\text{-literals} \\
& \leq \text{GAP} \\
& \leq \text{PATH}
\end{align*}
\]
A Decision Procedure for $\text{TSL}_K$

$\text{TSL}_K = \cdots \oplus T_{\text{Reachability}} \oplus \cdots \oplus \text{bridge functions and predicates}$

\[ \uparrow \quad \text{normalized } \text{TSL}_K\text{-literals} \quad \downarrow \]

$\widehat{\text{TSL}_K}$

\[ T_{\text{Base}} = T_{\text{addr}} \oplus T_{\text{elem}} \oplus T_{\text{cell}} \oplus T_{\text{mem}} \oplus T_{\text{fseq}} \oplus T_{\text{path}} \oplus T_{\text{set}} \oplus T_{\text{setth}} \oplus T_{\text{thid}} \oplus T_{\text{mrgn}} \oplus T_{\text{ord}} \oplus T_{\text{level}_K} \]

- By SMP it is always possible to enumerate all finitely many ground terms
A Decision Procedure for $\text{TSL}_K$

$$\text{TSL}_K = \cdots \oplus T_{\text{Reachability}} \oplus \cdots \oplus \text{bridge functions and predicates}$$

normalized $\text{TSL}_K$-literals

$\text{TSL}_K$

$T_{\text{Base}} = T_{\text{addr}} \oplus T_{\text{elem}} \oplus T_{\text{cell}} \oplus T_{\text{mem}} \oplus \underbrace{T_{\text{fseq}}}_\text{PATH} \oplus \underbrace{T_{\text{path}}}_\text{GAP} \oplus T_{\text{set}} \oplus T_{\text{setth}} \oplus T_{\text{thid}} \oplus T_{\text{mrgn}} \oplus T_{\text{ord}} \oplus T_{\text{level}_K}$

- By SMP it is always possible to enumerate all finitely many ground terms
- Unfolding of definitions in $\text{PATH}$ and $\text{GAP}$
A Decision Procedure for $\text{TSL}_K$

$$\text{TSL}_K = \cdots \oplus T_{\text{Reachability}} \oplus \cdots \oplus \text{bridge functions and predicates}$$

normalized $\text{TSL}_K$-literals

$T_{\text{Base}} = T_{\text{addr}} \oplus T_{\text{elem}} \oplus T_{\text{cell}} \oplus T_{\text{mem}} \oplus T_{\text{fseq}} \oplus T_{\text{set}} \oplus T_{\text{setth}} \oplus T_{\text{thid}} \oplus T_{\text{mrgn}} \oplus T_{\text{ord}} \oplus T_{\text{level}_K}$

- By SMP it is always possible to enumerate all finitely many ground terms
- Unfolding of definitions in $\text{PATH}$ and $\text{GAP}$
Conclusions

- A method to verify Concurrent Datastructures
- Thanks to Decision Procedures, automatic verification for
  - Concurrent Single Linked Lists
  - Concurrent Skiplists
- Future work
  - Other concurrent datastructures (trees, graphs...)
  - Implementation
- Many possible collaborations:
  - Decision procedures as combinations
  - Use of STM
Let $\Gamma$ be a conjunction of normalized $\text{TSL}_K$-literals. Let $\bar{e} = |V_{\text{elem}}(\Gamma)|$, $\bar{a} = |V_{\text{addr}}(\Gamma)|$, $\bar{m} = |V_{\text{mem}}(\Gamma)|$, $\bar{p} = |V_{\text{path}}(\Gamma)|$, $\bar{t} = |V_{\text{thid}}(\Gamma)|$ and $\bar{o} = |V_{\text{ord}}(\Gamma)|$. Then the following are equivalent:

- $\Gamma$ is $\text{TSL}_K$-satisfiable;
- $\Gamma$ is true in a $\text{TSL}_K$ interpretation $A$ such that

$$
|A_{\text{addr}}| \leq \bar{a} + 1 + \bar{m} \bar{a} + \bar{p}^2 + \bar{p}^3 + (K + 2)\bar{m}\bar{p}
$$

$$
|A_{\text{elem}}| \leq \bar{e} + \bar{m} |A_{\text{addr}}|
$$

$$
|A_{\text{thid}}| \leq \bar{k} + K\bar{m} |A_{\text{addr}}| + 1
$$

$$
|A_{\text{level}_K}| \leq K
$$

$$
|A_{\text{ord}}| \leq \bar{o} + \bar{m} |A_{\text{addr}}|
$$
Small Model Property

\[ L = \{ \eta_i \mid 1 \leq i \leq K \} \]

\[ O = V_{ord}^B \cup \{ m^B(v).key^B \mid m \in V_{mem} \text{ and } v \in X \} \]

\[ X = V_{addr}^B \cup \{ \text{null}^B \} \cup \{ m^B(v^B).next^B \mid m \in V_{mem} \text{ and } v \in V_{addr} \} \cup \{ v \in \delta(p^B, q^B) \mid \text{the literal } p \neq q \text{ is in } \Gamma \} \cup \{ v \in \sigma(p_1^B, p_2^B) \mid \text{the literal } \neg \text{append}(p_1, p_2, p_3) \text{ is in } \Gamma \text{ and } \text{path2set}^B(p_1^B) \cap \text{path2set}^B(p_2^B) \neq \emptyset \} \cup \{ v \in \sigma(p_1^B \circ p_2^B, p_3^B) \mid \text{the literal } \neg \text{append}(p_1, p_2, p_3) \text{ is in } \Gamma \text{ and } \text{path2set}^B(p_1^B) \cap \text{path2set}^B(p_2^B) = \emptyset \} \cup \{ v \in \kappa(m, p, l) \mid \text{firstlocked}(m, p, l) \text{ is in } \Gamma \} \cup \{ v \in \xi(m, p) \mid \neg \text{ordList}(m, p) \text{ is in } \Gamma \} \]

\[ Y = V_{thid}^B \cup \{ \emptyset \} \cup \{ m^B(v).lockid^B \mid m \in V_{mem} \text{ and } v \in X \} \]

\[ Z = V_{elem}^B \cup \{ m^B(v).data^B \mid m \in V_{mem} \text{ and } v \in X \} \]
**PATH**  *definitions*

\[\text{app} : \text{fseq} \times \text{fseq} \rightarrow \text{fseq}\]

\[
\begin{align*}
\text{app}(\text{nil}, l) &= l \\
\text{app}(\text{cons}(a, l), l'') &= \text{cons}(a, \text{app}(l, l''))
\end{align*}
\]

\[\text{fseq2set} : \text{fseq} \rightarrow \text{set}\]

\[
\begin{align*}
\text{fseq2set}(\text{nil}) &= \emptyset \\
\text{fseq2set}(\text{cons}(a, l)) &= \{a\} \cup \text{fseq2set}(l)
\end{align*}
\]

\[\text{ispath} : \text{fseq}\]

\[
\begin{align*}
\text{ispath}(\text{nil}) \\
\text{ispath}(\text{cons}(a, \text{nil})) \\
\{a\} \not\subseteq \text{fseq2set}(l) \land \text{ispath}(l) \rightarrow \text{ispath}(\text{cons}(a, l))
\end{align*}
\]

\[\text{last} : \text{fseq} \rightarrow \text{addr}\]

\[
\begin{align*}
\text{last}(\text{cons}(a, \text{nil})) &= a \\
l \neq \text{nil} \rightarrow \text{last}(\text{cons}(a, l)) &= \text{last}(l)
\end{align*}
\]

\[\text{isreachable} : \text{mem} \times \text{addr} \times \text{addr}\]

\[
\begin{align*}
\text{isreachable}(m, a, a) \\
m[a].\text{next} = a' \land \text{isreachable}(m, a', b) \rightarrow \text{isreachable}(m, a, b)
\end{align*}
\]

\[\text{isreachablep} : \text{mem} \times \text{addr} \times \text{addr} \times \text{fseq}\]

\[
\begin{align*}
\text{isreachablep}(m, a, a, \text{nil}) \\
m[a].\text{next} = a' \land \text{isreachablep}(m, a', b, p) \rightarrow \text{isreachablep}(m, a, b, \text{cons}(a, p))
\end{align*}
\]

\[\text{firstmarked} : \text{mem} \times \text{fseq} \times \text{addr}\]

\[
\begin{align*}
\text{firstmarked}(m, \text{nil}, \text{null}) \\
p \neq \text{nil} \land p = \text{cons}(j, q) \land m[j].\text{lockid} \neq \emptyset \rightarrow \text{firstmarked}(m, p, j) \\
p \neq \text{nil} \land p = \text{cons}(j, q) \land m[j].\text{lockid} = \emptyset \land \text{firstmarked}(m, q, i) \rightarrow \text{firstmarked}(m, p, i)
\end{align*}
\]
GAP definitions

\[
nil = \epsilon
\]

\[
\text{cons}(a, \text{nil}) = [a]
\]

\[
\begin{align*}
\text{ispath}(p_1) \land \text{ispath}(p_2) \land \\
\text{fseq2set}(p_1) \cap \text{fseq2set}(p_2) = \emptyset \land \\
\text{app}(p_1, p_2) = p_3
\end{align*}
\]

\[
\leftrightarrow \text{append}(p_1, p_2, p_3)
\]

\[
\text{ispath}(p) \rightarrow \text{isreachable}_K (m, a, b, l, p) = \text{reach}_K (m, a, b, l, p)
\]

\[
\text{ispath}(p) \rightarrow \text{fseq2set}(p) = \text{path2set}(p)
\]

\[
\text{isreachable}_K (m, a, b, l, p) \rightarrow \text{getp}_K (m, a, b, l) = p
\]

\[
\neg \text{isreachable}_K (m, a, b, l, p) \rightarrow \text{getp}_K (m, a, b, l) = \text{nil}
\]

\[
\text{ispath}(p) \land \text{firstmarked}(m, p, l, i) \leftrightarrow \text{firstlocked}(m, p, l) = i
\]

\[
\text{ispath}(p) \land \text{ordPath}(m, p) \leftrightarrow \text{ordList}(m, p)
\]