Decision Procedures for Concurrent Skiplists

Alejandro Sánchez[1] César Sánchez[1,2]

[1]The IMDEA Software Institute, Spain

[2]Spanish Council for Scientific Research (CSIC), Spain

EPFL, Lausanne, 16 September 2010

Why do we want a decision procedure

- Imperative programs

P

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

$$\frac{P_1||P_2||\cdots||P_n}{\downarrow}$$

data structures
(heap)

Why do we want a decision procedure

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

$$\underbrace{P_1||P_2||\cdots||P_n}_{\downarrow} \models \varphi$$

data structures
(heap)

Why do we want a decision procedure

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

$$\underline{P_1||P_2||\cdots||P_n} \models \varphi$$

Regional Logic

data structures (heap)

LTL ($\Box$, $\Diamond$, $\mathcal{U}$, ...)

Why do we want a decision procedure

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

Verification Diagrams

$$\underline{P_1||P_2||\cdots||P_n} \models \varphi$$

$\downarrow$

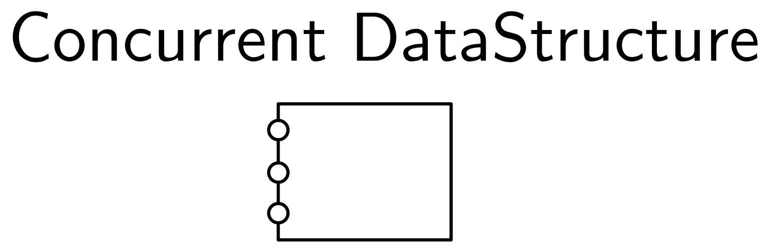
Regional Logic

data structures (heap)

LTL ($\square, \lozenge, \mathcal{U}, \ldots$)

Verification of Concurrent Data-structures

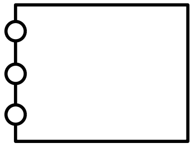
Main Idea



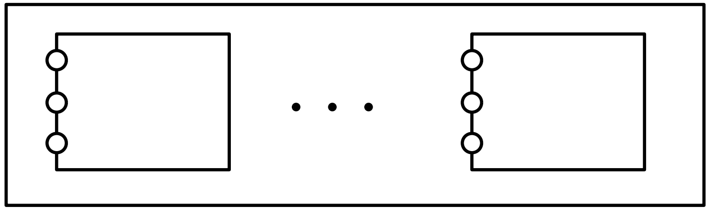
Verification of Concurrent Data-structures

Main Idea

Concurrent DataStructure



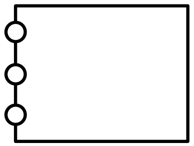
Most General Client



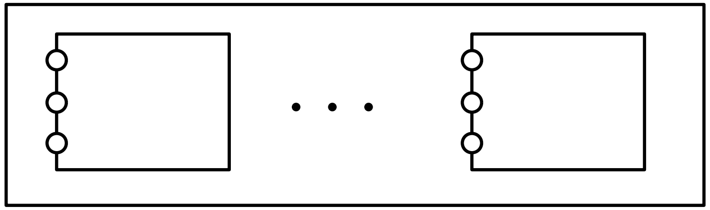
Verification of Concurrent Data-structures

Main Idea

Concurrent DataStructure



Most General Client

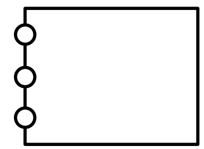


$$P[N] : P(1) || \cdots || P(N)$$

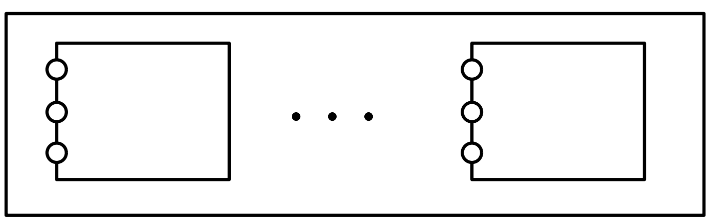
Verification of Concurrent Data-structures

Main Idea

Concurrent DataStructure



Most General Client



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

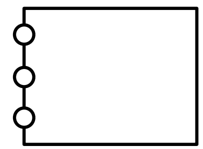
$$+$$

ghost variables

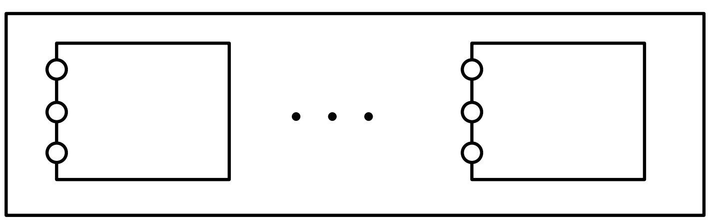
Verification of Concurrent Data-structures

Main Idea

Concurrent DataStructure



Most General Client



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

$$+$$

ghost variables

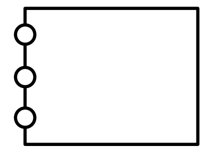
Property

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

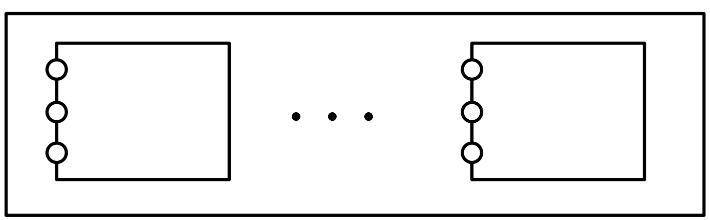
Verification of Concurrent Data-structures

Main Idea

Concurrent DataStructure



Most General Client



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

$$+$$

ghost variables

Diagram

$$\mathcal{D}$$

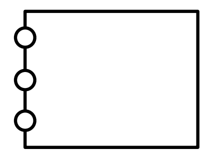
Property

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

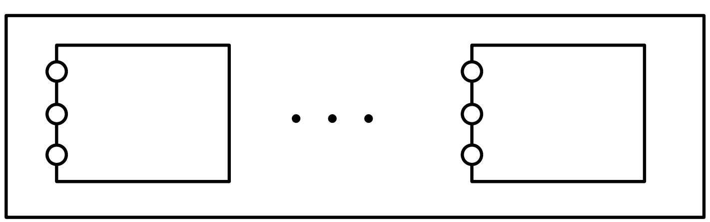
Verification of Concurrent Data-structures

Main Idea

Concurrent DataStructure



Most General Client



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

$+$

ghost variables

Diagram

Property

$$\models \quad \mathcal{D} \quad \models \quad \varphi^{(k)}$$

Verification Conditions:

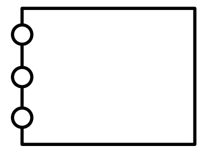
- Initiation
- Consecution
- Acceptance
- Fairness

Satisfaction (Model Checking)

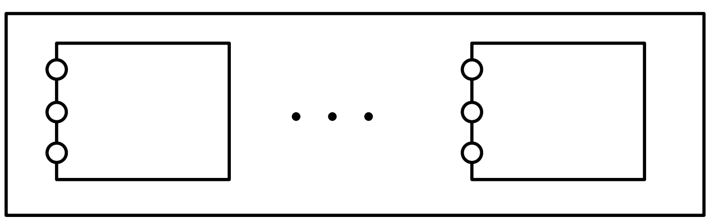
Verification of Concurrent Data-structures

Main Idea

Concurrent DataStructure



Most General Client



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

$+$

ghost variables

Diagram

Property

$\models \quad \mathcal{D} \quad \models \quad \varphi^{(k)}$

Verification Conditions:

- Initiation
- Consecution
- Acceptance
- Fairness

Decision Procedures

Satisfaction (Model Checking)

Verification Conditions

Verification Conditions

- *Initiation*

$$\Theta \longrightarrow \mu(N_0)$$

Verification Conditions

- *Initiation*

  $$\Theta \to \mu(N_0)$$

- *Consecution*: for all n and τ:

  $$\mu(n)(s) \wedge \rho_\tau(s, s') \to \mu(\textit{next}(n))(s')$$

Verification Conditions

- *Initiation*

  $$\Theta \rightarrow \mu(N_0)$$

- *Consecution*: for all n and τ:

  $$\mu(n)(s) \wedge \rho_\tau(s, s') \rightarrow \mu(\mathit{next}(n))(s')$$

- *Acceptance*: if $(n_1, n_2) \in P \setminus R$ then

  $$\mu(n_1)(s) \wedge \mu(n_2)(s') \wedge \rho_\tau(s, s') \rightarrow \delta_{n_1}(s) \geq \delta_{n_2}(s')$$

  and if $(n_1, n_2) \notin P \cup R$:

  $$\mu(n_1)(s) \wedge \mu(n_2)(s') \wedge \rho_\tau(s, s') \rightarrow \delta_{n_1}(s) > \delta_{n_2}(s')$$

Verification Conditions

- *Initiation*

$$\Theta \rightarrow \mu(N_0)$$

- *Consecution*: for all n and τ:

$$\mu(n)(s) \wedge \rho_\tau(s, s') \rightarrow \mu(\textit{next}(n))(s')$$

- *Acceptance*: if $(n_1, n_2) \in P \setminus R$ then

$$\mu(n_1)(s) \wedge \mu(n_2)(s') \wedge \rho_\tau(s, s') \rightarrow \delta_{n_1}(s) \geq \delta_{n_2}(s')$$

  and if $(n_1, n_2) \notin P \cup R$:

$$\mu(n_1)(s) \wedge \mu(n_2)(s') \wedge \rho_\tau(s, s') \rightarrow \delta_{n_1}(s) > \delta_{n_2}(s')$$

- *Fairness*: for all n and $\tau \in \eta(n, n')$:

$$\mu(n)(s) \rightarrow En_\tau(s)$$

$$\mu(n)(s) \wedge \rho_\tau(s, s') \rightarrow \mu(\tau(n))(s')$$

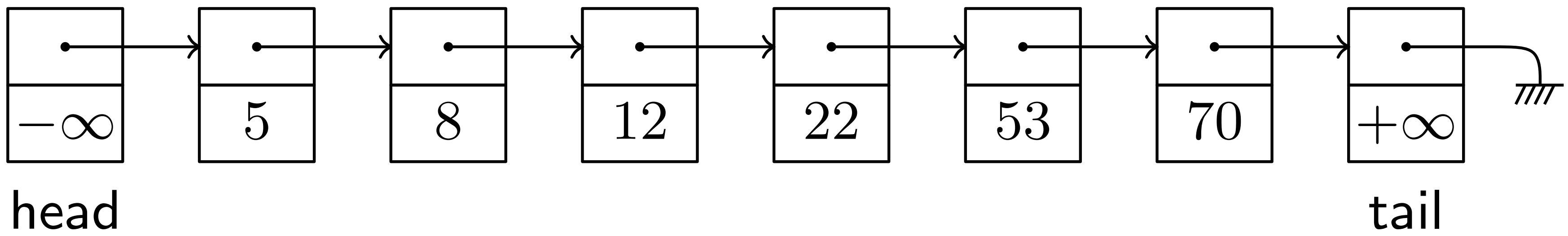
Concurrent Lock-Coupling Skiplists

Concurrent Lock-Coupling Skiplists

- Sorted list of elements

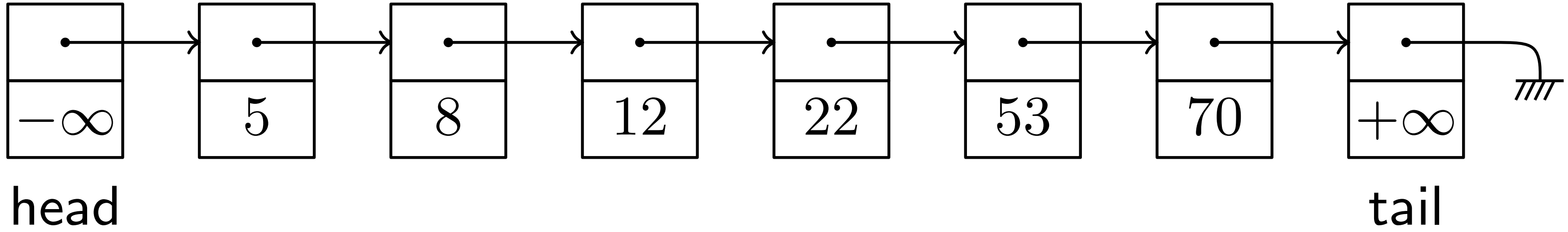
Concurrent Lock-Coupling Skiplists

- Sorted list of elements



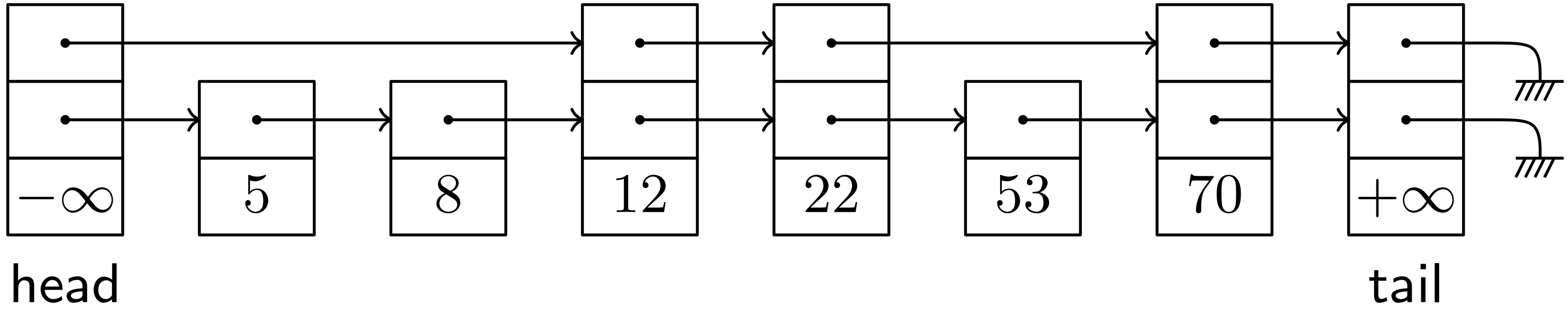
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists



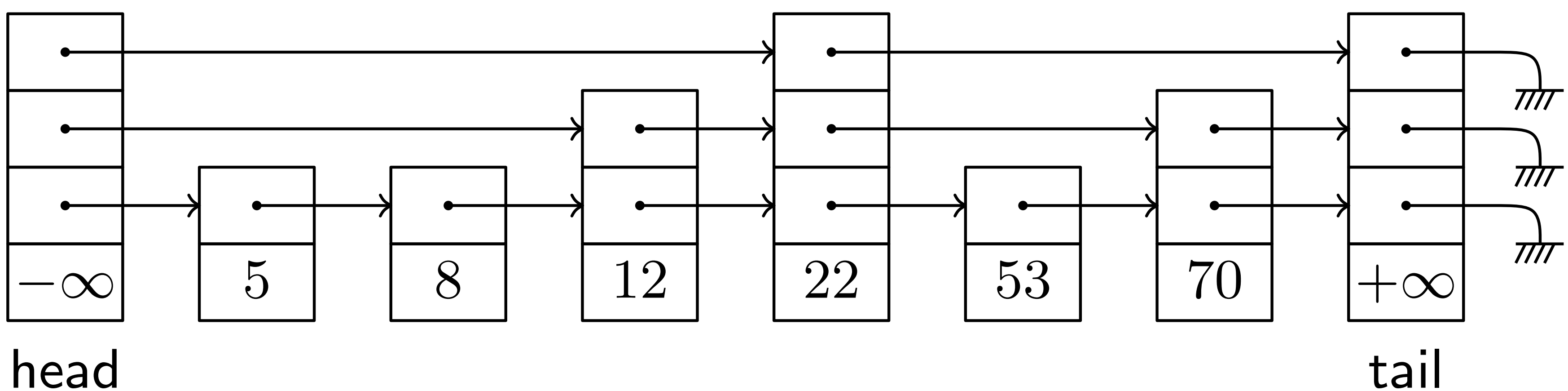
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists



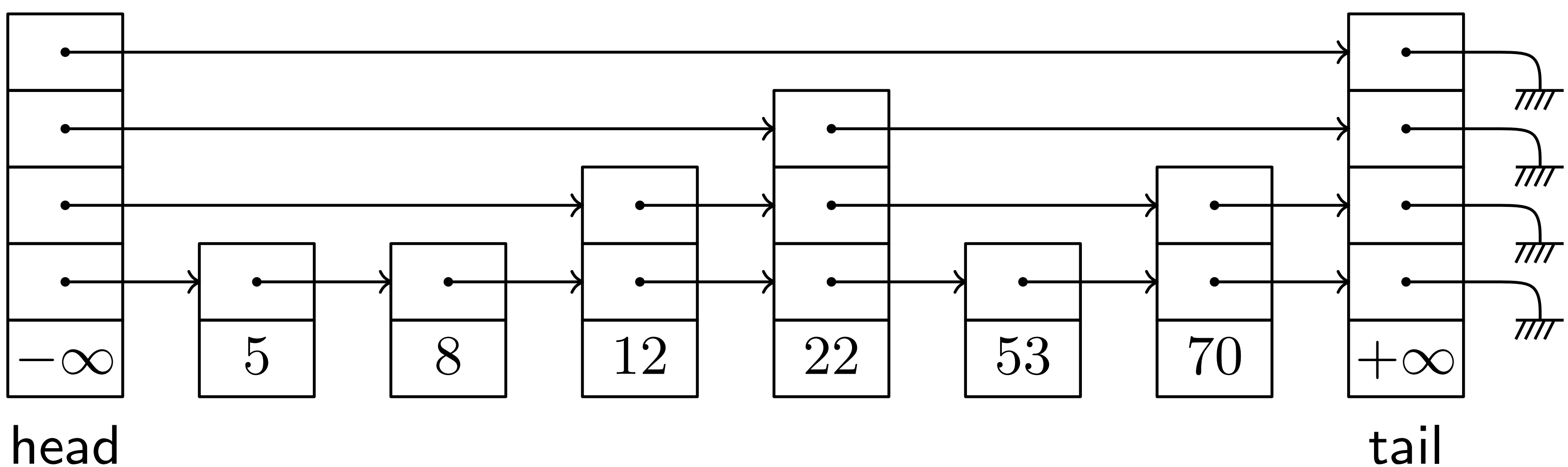
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists



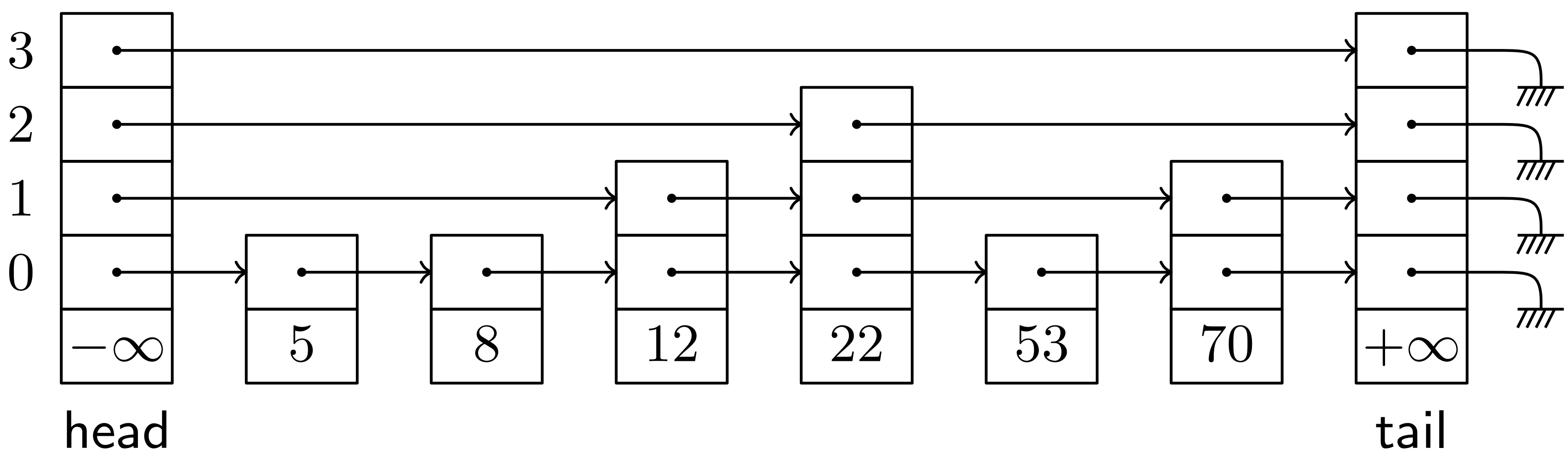
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists



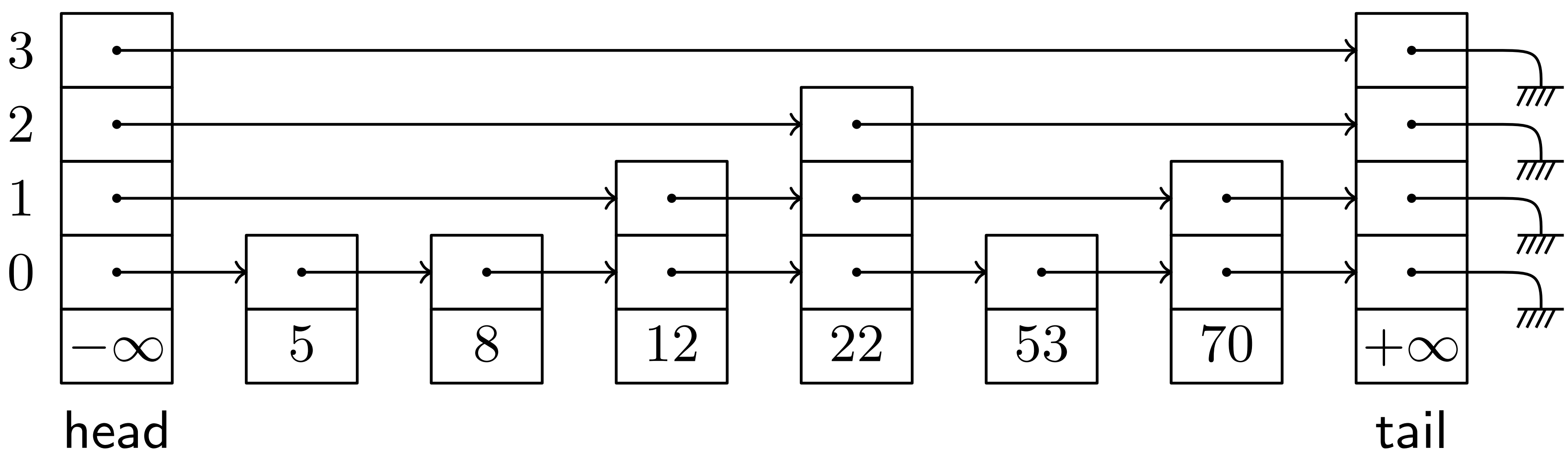
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists



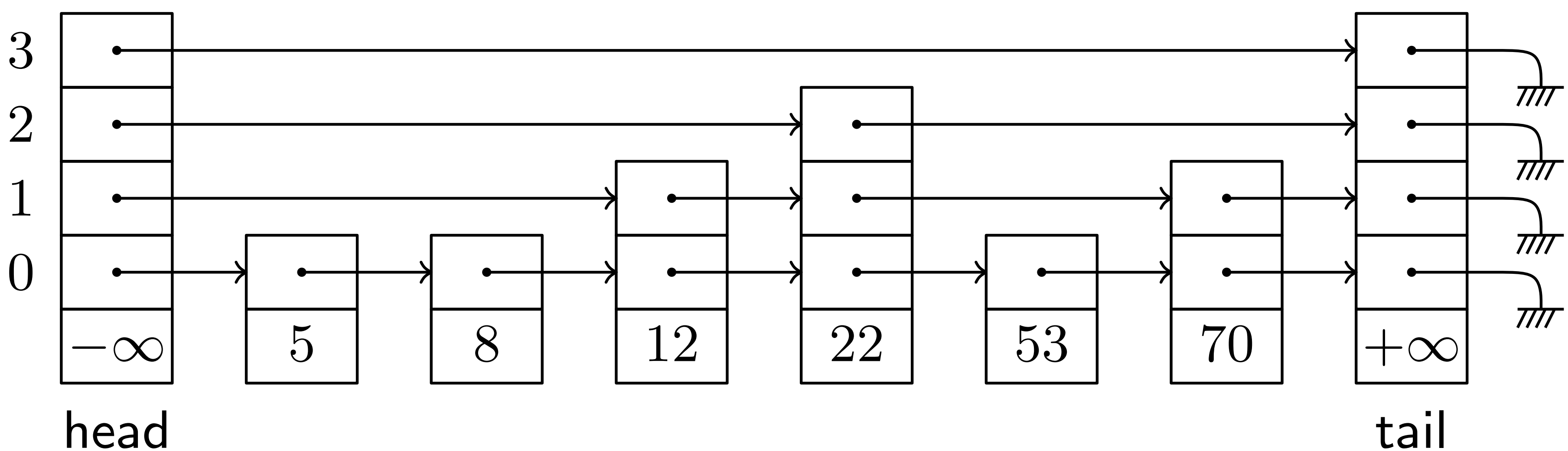
Concurrent Lock-Coupling Skiplists

- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees



Concurrent Lock-Coupling Skiplists

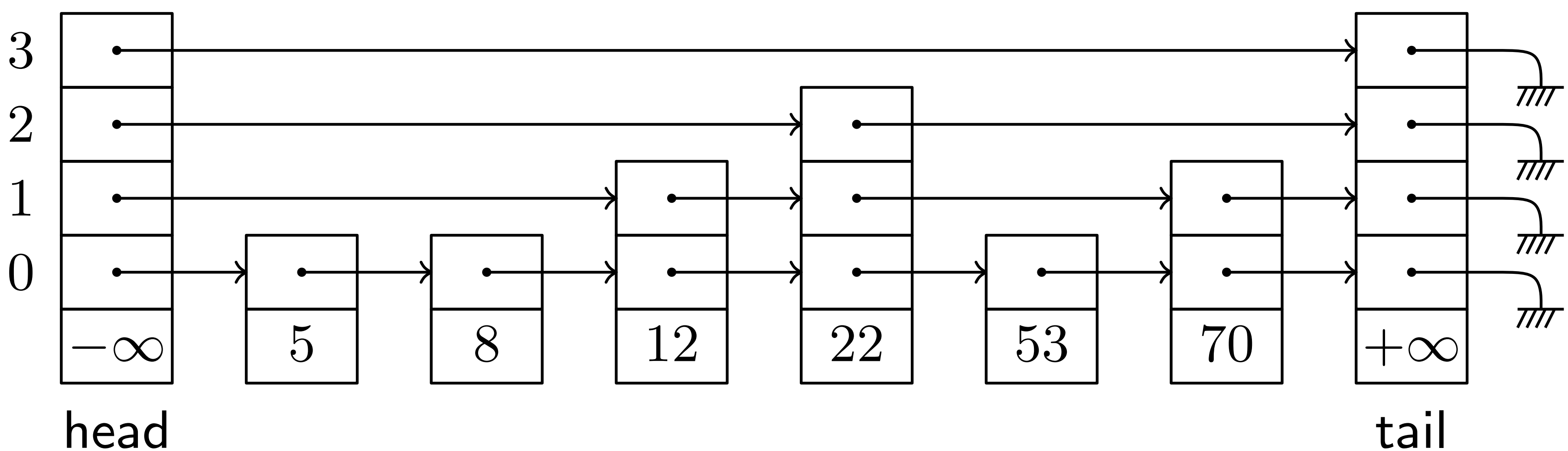
- Sorted list of elements
- Hierarchy of linked lists
- Efficiency comparable to balanced binary search trees
- Reduce granularity of locks (in climbing fashion)



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)

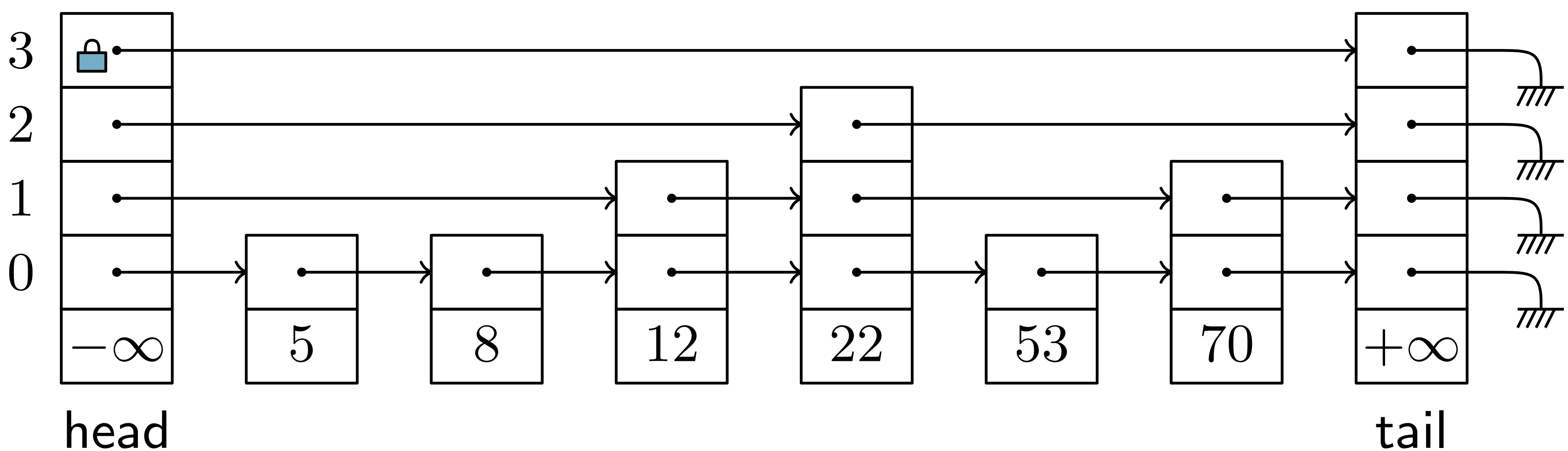
insert(60) with height 1



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)

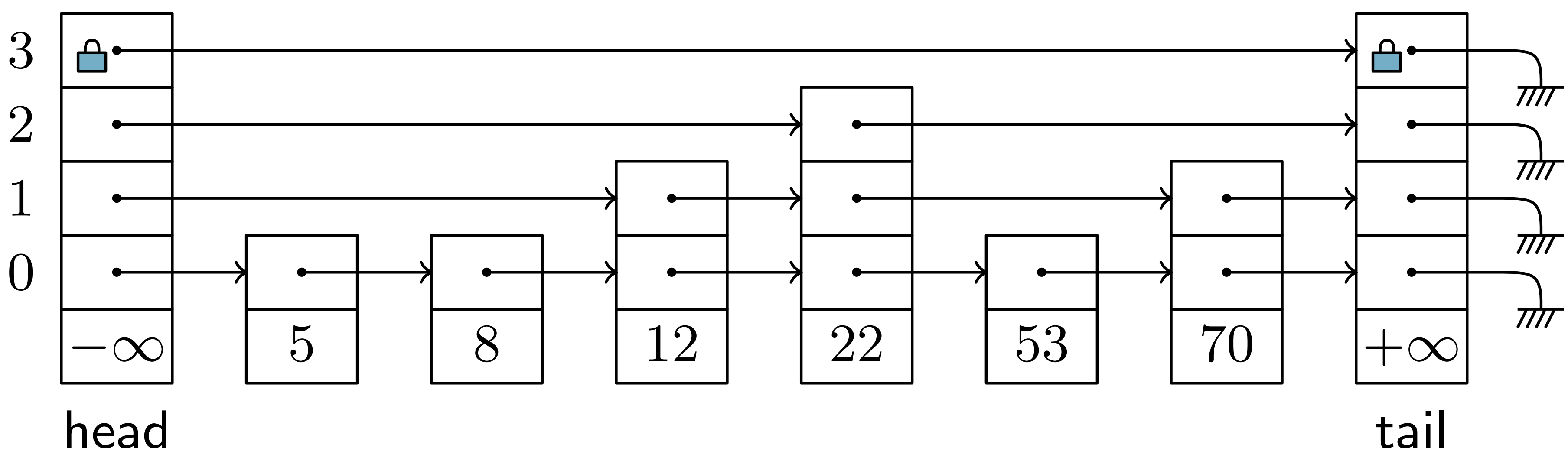
insert(60) with height 1



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)

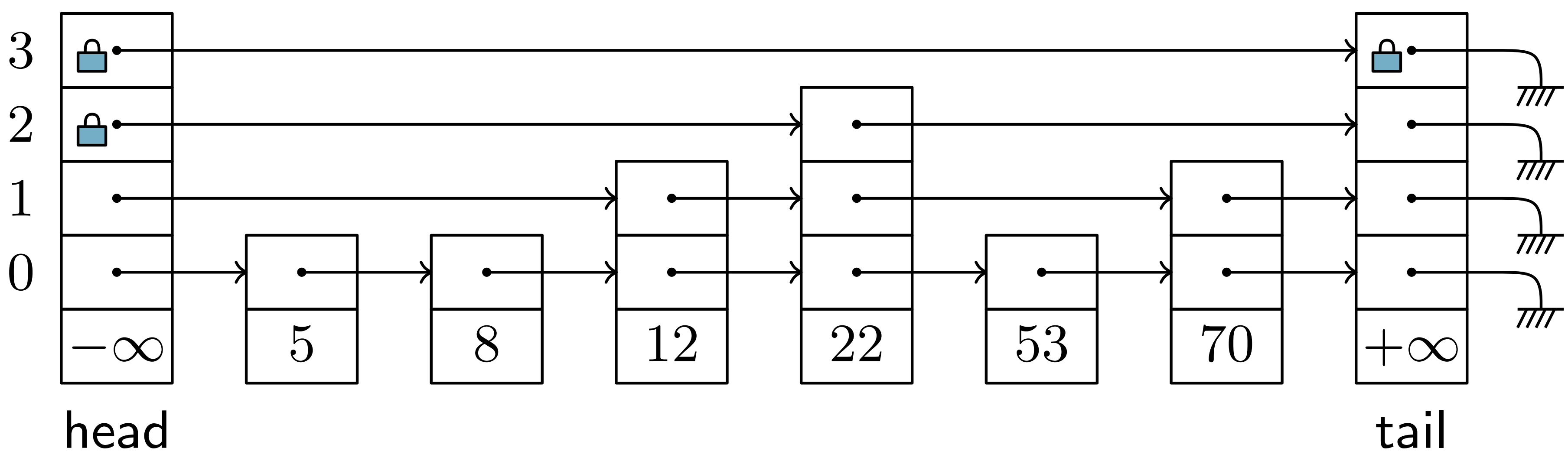
insert(60) with height 1



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)

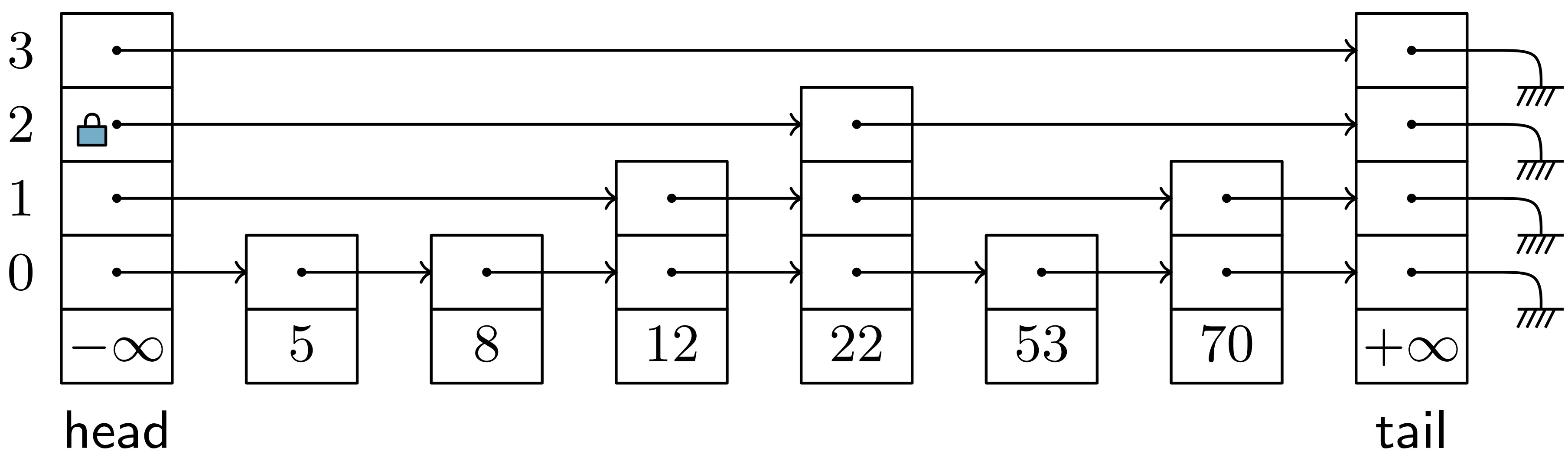
insert(60) with height 1



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)

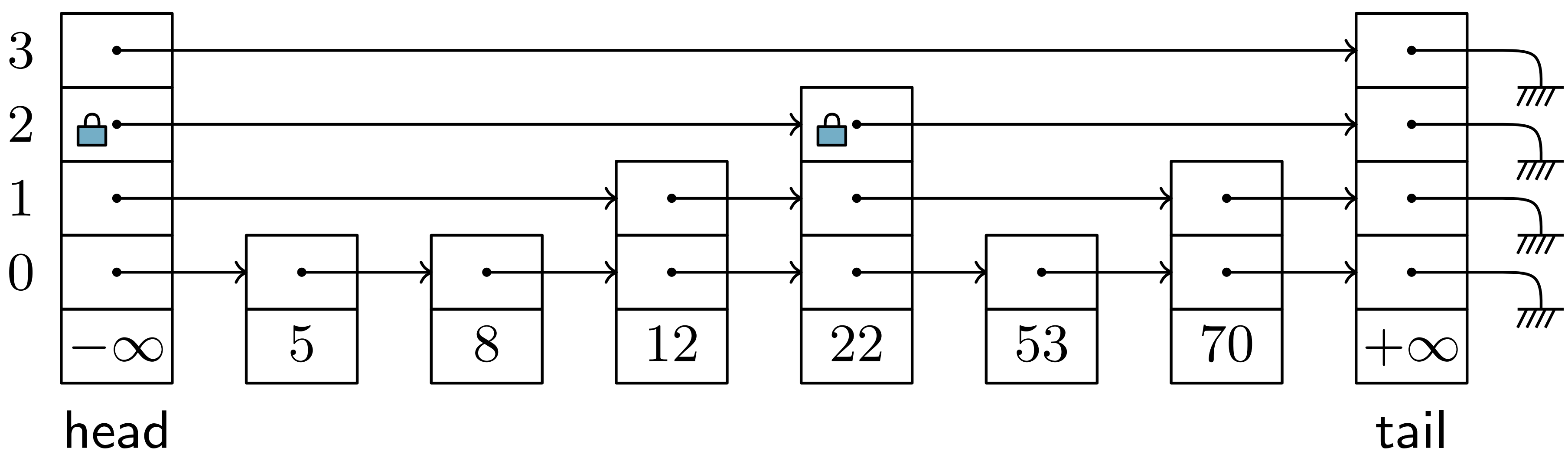
insert(60) with height 1



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)

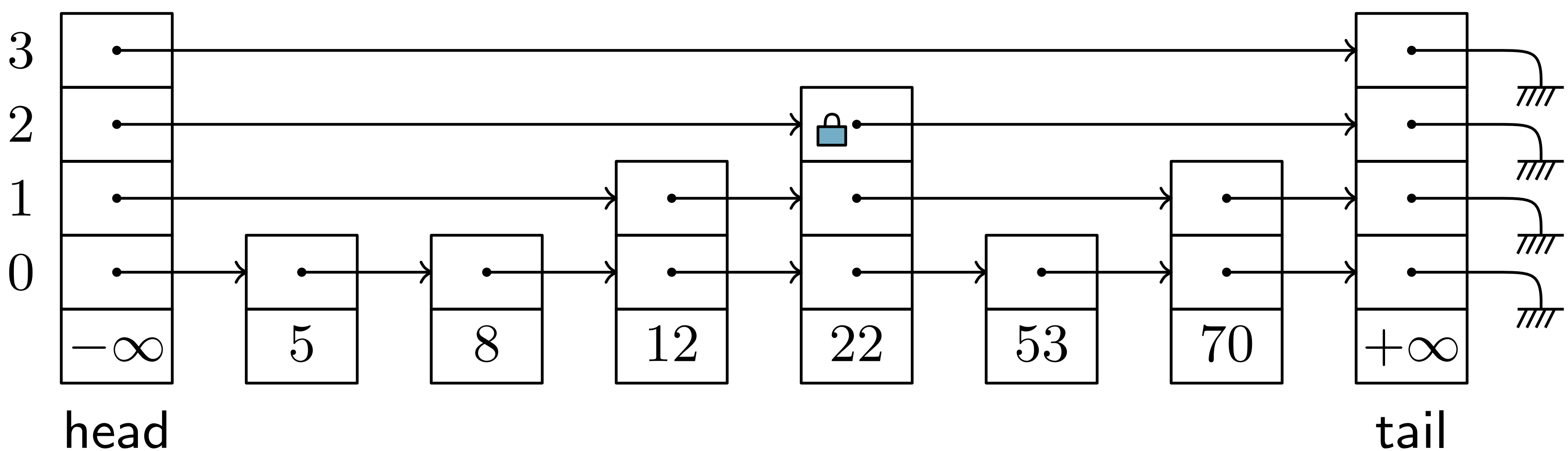
insert(60) with height 1



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)

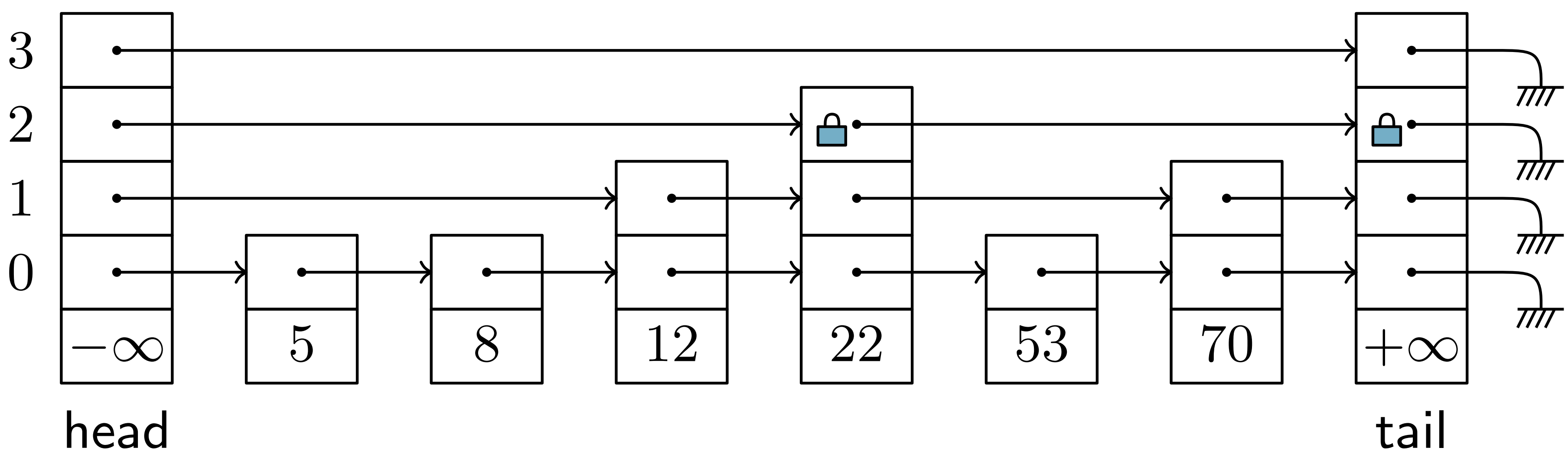
insert(60) with height 1



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)

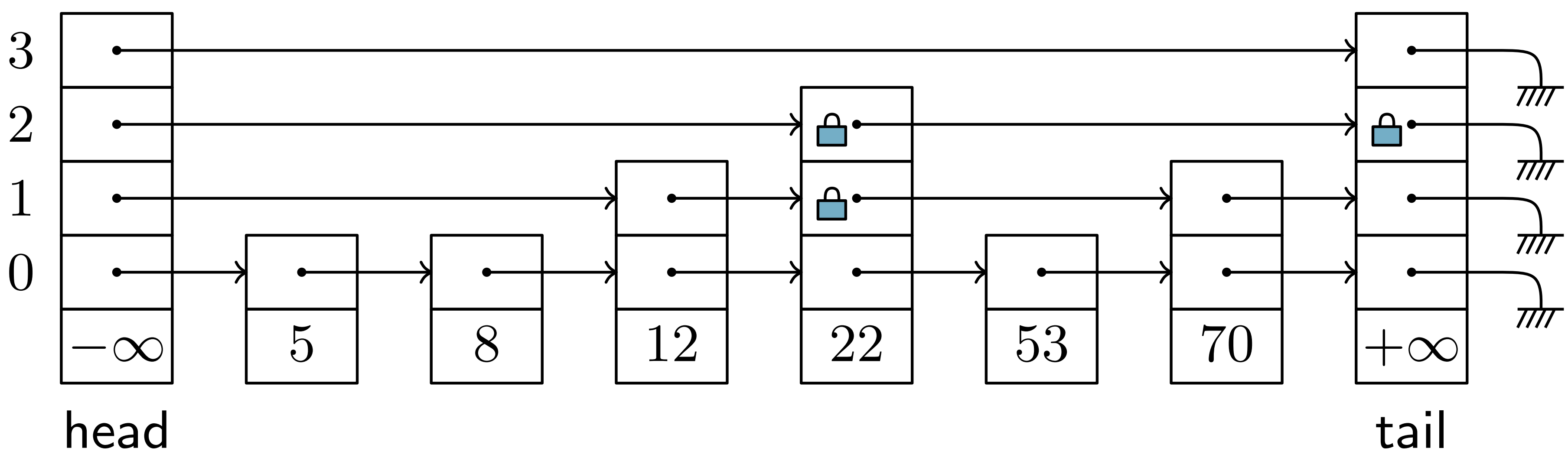
insert(60) with height 1



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)

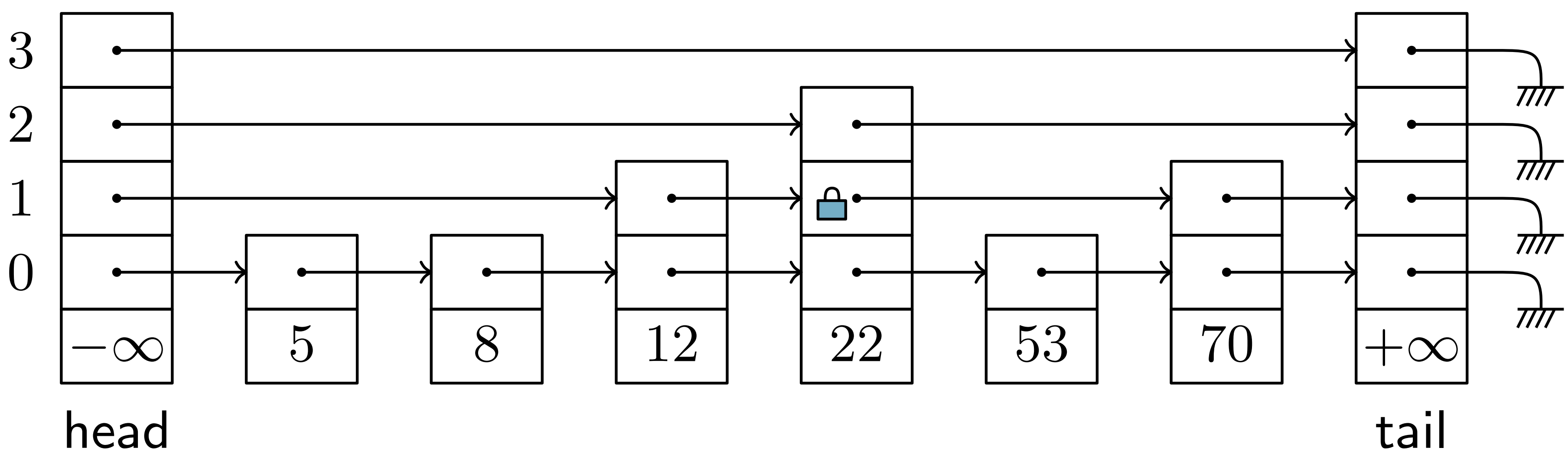
insert(60) with height 1



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)

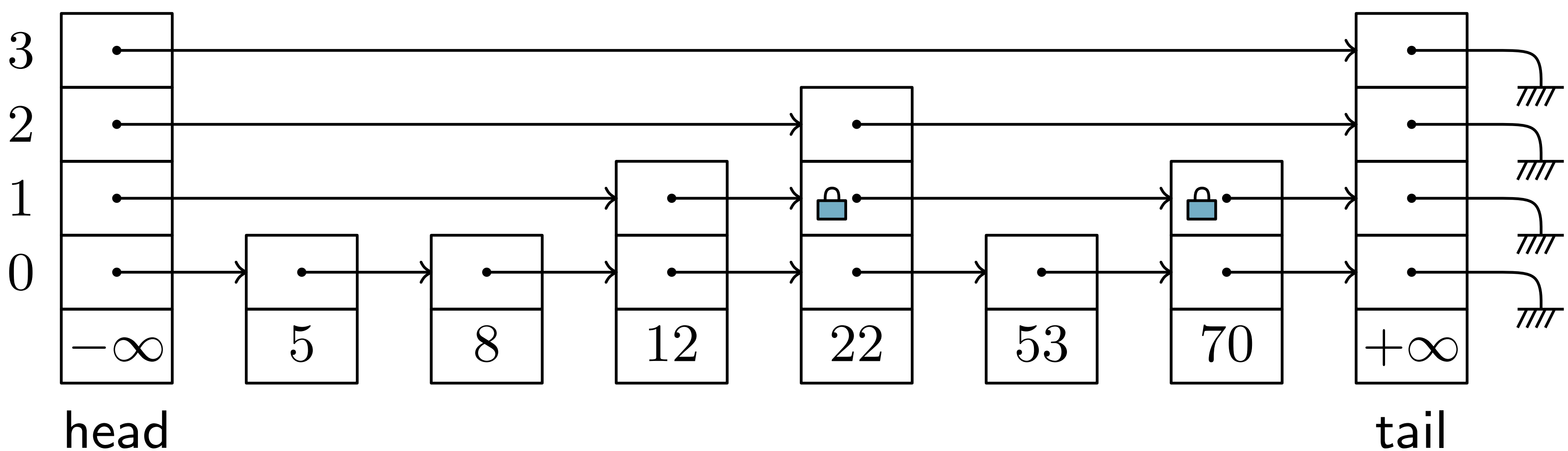
insert(60) with height 1



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)

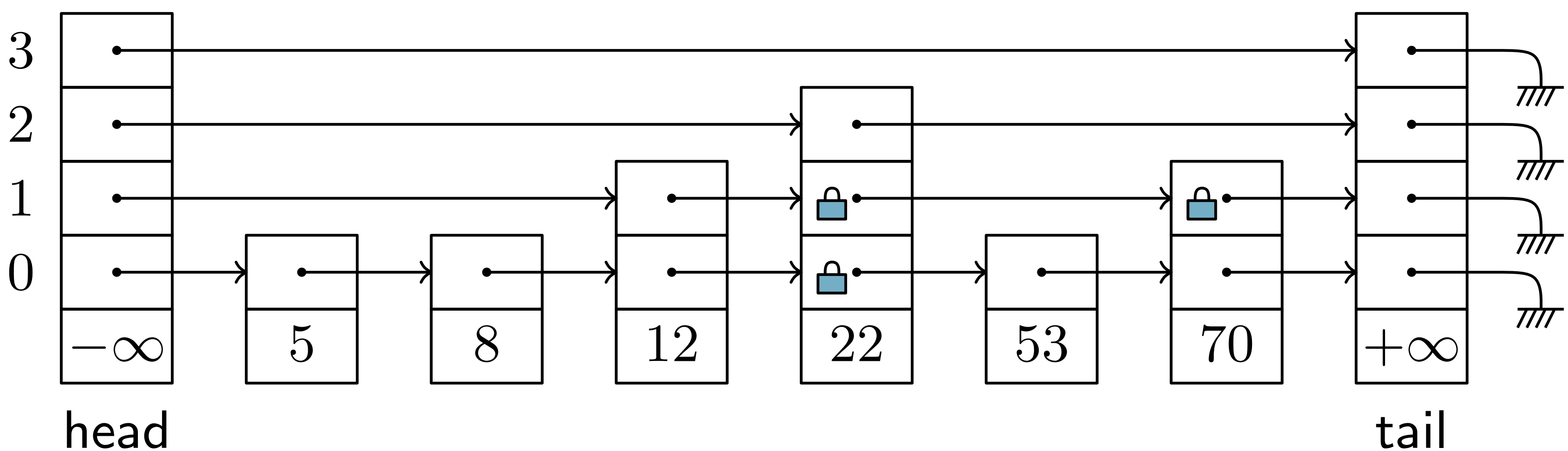
insert(60) with height 1



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)

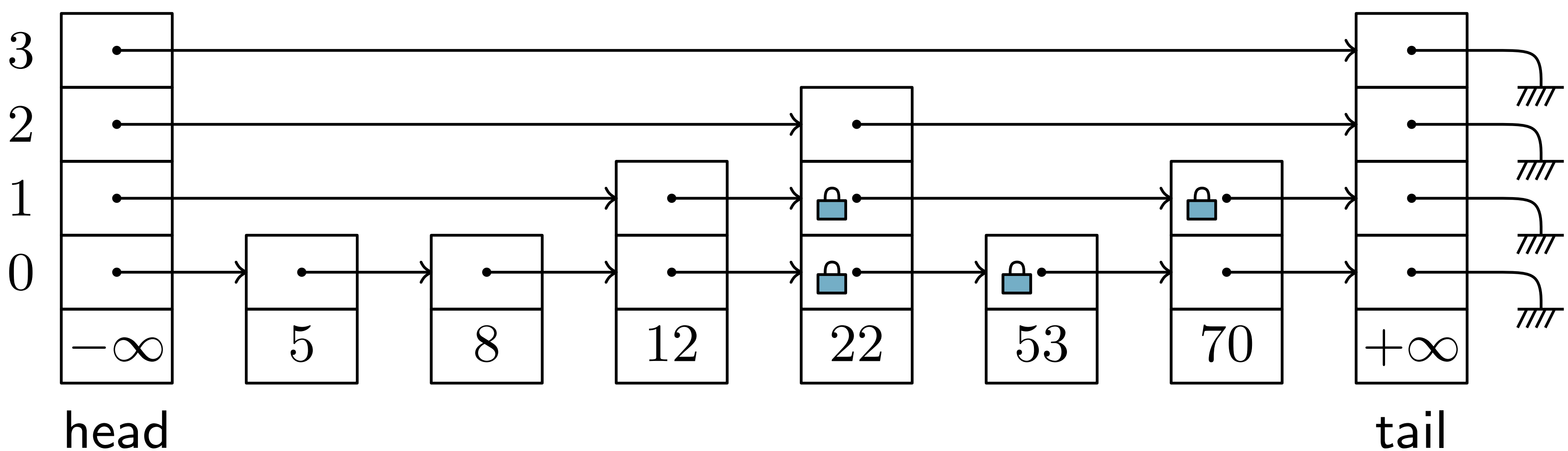
insert(60) with height 1



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)

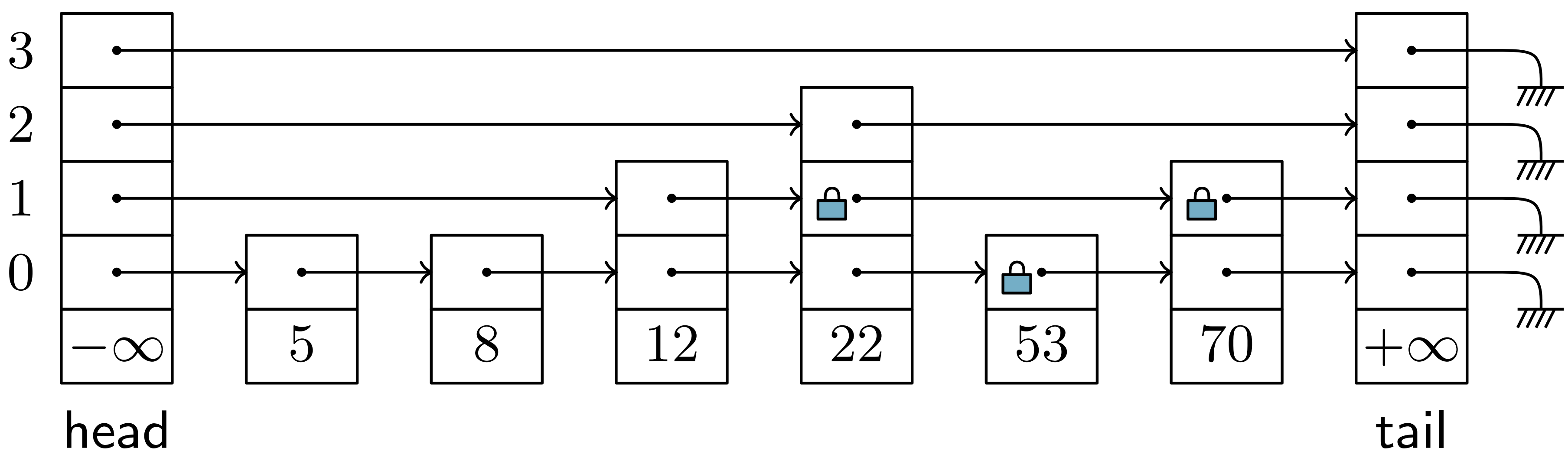
insert(60) with height 1



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)

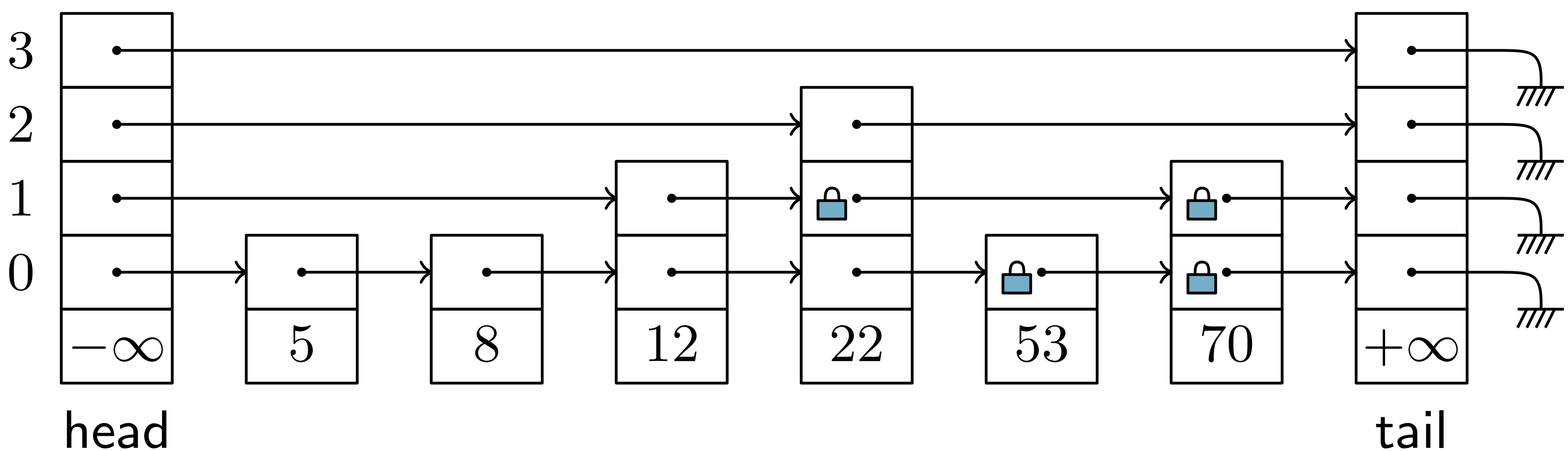
insert(60) with height 1



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)

insert(60) with height 1



Verification Conditions Examples

Verification Conditions Examples

- Preservation of skiplistness shape

Verification Conditions Examples

- Preservation of skiplistness shape

$$
\begin{aligned}
&SkipList_3(sl : SkipList) \mathrel{\hat{=}} \\
&\qquad OList_0(h, sl.head, sl.r_0) \wedge \\
&\qquad OList_1(h, sl.head, sl.r_1) \wedge \\
&\qquad OList_2(h, sl.head, sl.r_2) \wedge \\
&\qquad \pi_1(sl.r_3) \subseteq \pi_1(sl.r_2) \subseteq \pi_1(sl.r_1) \wedge \\
&\qquad sl.last.next_0 = null \wedge sl.last.next_1 = null \wedge \\
&\qquad sl.last.next_2 = null \wedge \\
&\qquad SubPath(getp_1(h, sl.head, sl.last), getp_0(h, sl.head, sl.last)) \wedge \\
&\qquad SubPath(getp_2(h, sl.head, sl.last), getp_1(h, sl.head, sl.last))
\end{aligned}
$$

Verification Conditions Examples

- Preservation of skiplistness shape
- Program transitions

Verification Conditions Examples

- Preservation of skiplistness shape
- Program transitions

$$
\begin{aligned}
&SkipList_3(sl) \wedge at\_insert_{31} \wedge 0 \leq i \leq 2 \wedge \\
&x.val = v \wedge update[i].val < v \wedge \\
&update[i].next[i].val > v \wedge x.next[i] = update[i].next[i] \wedge \\
&m_r = \{(update[i], i), (x.next[i], i)\} \cup m_{i+1..2} \wedge update[i].locks[i] = t \wedge \\
&update[i].next[i].locks[i] = t \wedge (j < i \rightarrow (x, i) \in sl.r_j) \wedge \\
&update'[i].next[i] := x \wedge sl'.r_i := sl.r_i \cup \{(x, i)\} \rightarrow \\
&\qquad SkipList_3(sl') \wedge at'\_insert_{32} \wedge update'[i].key < k \wedge \\
&\qquad update'[i].next[i].next[i]key > k \wedge \\
&\qquad x'.next[i] = update'[i].next[i].next[i] \wedge \\
&\qquad update'[i].next[i] = x' \wedge \\
&\qquad m'_r = \{(update'[i], i), (x'.next[i], i)\} \cup m'_{i+1..2} \wedge \\
&\qquad update'[i].locks[i] = t \wedge update'[i].next[i].next[i].locks[i] = t
\end{aligned}
$$

Theory of Concurrent Skiplists of Height K (TSL_K)

Theory of Concurrent Skiplists of Height K (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 ($\mathsf{TSL_K}$)

- 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 ($\mathsf{TSL_K}$)

- Union of theories

$$\mathsf{TSL_K} \quad = \quad T_{\mathsf{addr}}$$

$$\Sigma_{\mathsf{addr}} = \ \{\mathsf{addr}\}, \emptyset, \emptyset$$

Theory of Concurrent Skiplists of Height K (TSL_K)

- Union of theories

$$\mathsf{TSL}_\mathsf{K} \quad = \quad T_{\mathsf{addr}} \oplus T_{\mathsf{elem}}$$

$$\Sigma_{\mathsf{elem}} = \ \{\mathsf{elem}\}, \emptyset, \emptyset$$

Theory of Concurrent Skiplists of Height K ($\mathsf{TSL_K}$)

- Union of theories

$$\mathsf{TSL_K} \quad = \quad T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}}$$

$$\Sigma_{\mathsf{cell}} = \begin{array}{l} \{\mathsf{cell}, \mathsf{elem}, \mathsf{ord}, \mathsf{addr}, \mathsf{thid}\} \\ \left\{ \begin{array}{lcl} error & : & \mathsf{cell} \\ mkcell & : & \mathsf{elem} \times \mathsf{ord} \times \mathsf{addr}^{\mathsf{K}} \times \mathsf{thid}^{\mathsf{K}} \to \mathsf{cell} \\ \_.data & : & \mathsf{cell} \to \mathsf{elem} \\ \_.key & : & \mathsf{cell} \to \mathsf{ord} \\ \_.next[\_] & : & \mathsf{cell} \times \mathsf{level_K} \to \mathsf{addr} \\ \_.lockid[\_] & : & \mathsf{cell} \times \mathsf{level_K} \to \mathsf{thid} \\ \_.lock[\_] & : & \mathsf{cell} \times \mathsf{level_K} \to \mathsf{thid} \to \mathsf{cell} \\ \_.unlock[\_] & : & \mathsf{cell} \times \mathsf{level_K} \to \mathsf{cell} \end{array} \right\} \\ \emptyset \end{array}$$

Theory of Concurrent Skiplists of Height K ($\mathsf{TSL_K}$)

- Union of theories

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

$$\Sigma_{\mathsf{mem}} = \begin{array}{l} \{\mathsf{mem}, \mathsf{addr}, \mathsf{cell}\} \\ \left\{ \begin{array}{lcl} null & : & \mathsf{addr} \\ \_[\_] & : & \mathsf{mem} \times \mathsf{addr} \rightarrow \mathsf{cell} \\ upd & : & \mathsf{mem} \times \mathsf{addr} \times \mathsf{cell} \rightarrow \mathsf{mem} \end{array} \right\} \\ \emptyset \end{array}$$

Theory of Concurrent Skiplists of Height K ($\mathsf{TSL_K}$)

- Union of theories

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

$$\Sigma_{\mathsf{Reachability}} = \begin{array}{l} \{\mathsf{mem}, \mathsf{addr}, \mathsf{path}\} \\ \left\{ \begin{array}{lcl} \epsilon & : & \mathsf{path} \\ [\_] & : & \mathsf{addr} \rightarrow \mathsf{path} \end{array} \right\} \\ \left\{ \begin{array}{lcl} append & : & \mathsf{path} \times \mathsf{path} \times \mathsf{path} \\ reach_{\mathsf{K}} & : & \mathsf{mem} \times \mathsf{addr} \times \mathsf{addr} \times \mathsf{level_K} \times \mathsf{path} \end{array} \right\} \end{array}$$

Theory of Concurrent Skiplists of Height K ($\mathsf{TSL_K}$)

- Union of theories

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

$$\Sigma_{\mathsf{set}} = \begin{array}{l} \{\mathsf{addr}, \mathsf{set}\} \\ \left\{ \begin{array}{lcl} \emptyset & : & \mathsf{set} \\ \{\_\} & : & \mathsf{addr} \to \mathsf{set} \\ \cup, \cap, \setminus & : & \mathsf{set} \times \mathsf{set} \to \mathsf{set} \end{array} \right\} \\ \left\{ \begin{array}{lcl} \in & : & \mathsf{addr} \times \mathsf{set} \\ \subseteq & : & \mathsf{set} \times \mathsf{set} \end{array} \right\} \end{array}$$

Theory of Concurrent Skiplists of Height K ($\mathsf{TSL_K}$)

- Union of theories

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

$$\Sigma_{\mathsf{setth}} = \begin{array}{l} \{\mathsf{thid}, \mathsf{setth}\} \\ \left\{ \begin{array}{lcl} \emptyset_T & : & \mathsf{setth} \\ \{\_\}_T & : & \mathsf{thid} \to \mathsf{setth} \\ \cup_T, \cap_T, \setminus_T & : & \mathsf{setth} \times \mathsf{setth} \to \mathsf{setth} \end{array} \right\} \\ \left\{ \begin{array}{lcl} \in_T & : & \mathsf{thid} \times \mathsf{setth} \\ \subseteq_T & : & \mathsf{setth} \times \mathsf{setth} \end{array} \right\} \end{array}$$

Theory of Concurrent Skiplists of Height K ($\mathsf{TSL_K}$)

- Union of theories

$$\begin{aligned}\mathsf{TSL_K} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus T_{\mathsf{Reachability}} \oplus \\ & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}}\end{aligned}$$

$$\Sigma_{\mathsf{setth}} = \left\{\mathsf{thid}\right\}, \emptyset, \emptyset$$

Theory of Concurrent Skiplists of Height K ($\mathsf{TSL_K}$)

- Union of theories

$$\begin{aligned}\mathsf{TSL_K} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus T_{\mathsf{Reachability}} \oplus \\ & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}}\end{aligned}$$

$$\Sigma_{\mathsf{mrgn}} = \begin{array}{l} \{\mathsf{mrgn}, \mathsf{addr}, \mathsf{level_K}\} \\ \left\{ \begin{array}{lcl} \mathbf{emp}_{\mathsf{mr}} & : & \mathsf{mrgn} \\ \langle \_, \_ \rangle_{\mathsf{mr}} & : & \mathsf{addr} \times \mathsf{level_K} \to \mathsf{mrgn} \\ \cup_{\mathsf{mr}}, \cap_{\mathsf{mr}}, -_{\mathsf{mr}} & : & \mathsf{mrgn} \times \mathsf{mrgn} \to \mathsf{mrgn} \end{array} \right\} \\ \left\{ \begin{array}{lcl} \in_{\mathsf{mr}} & : & \mathsf{addr} \times \mathsf{level_K} \times \mathsf{mrgn} \\ \subseteq_{\mathsf{mr}} & : & \mathsf{mrgn} \times \mathsf{mrgn} \\ \#_{\mathsf{mr}} & : & \mathsf{mrgn} \times \mathsf{mrgn} \end{array} \right\} \end{array}$$

Theory of Concurrent Skiplists of Height K ($\mathsf{TSL_K}$)

- Union of theories

$$\begin{aligned}\mathsf{TSL_K} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus T_{\mathsf{Reachability}} \oplus \\ & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}}\end{aligned}$$

$$\Sigma_{\mathsf{ord}} = \begin{array}{l} \{\mathsf{ord}\} \\ \left\{ \begin{array}{lcl} 0 & : & \mathsf{ord} \\ +\infty & : & \mathsf{ord} \end{array} \right\} \\ \left\{ \begin{array}{lcl} \preceq & : & \mathsf{ord} \times \mathsf{ord} \end{array} \right\} \end{array}$$

Theory of Concurrent Skiplists of Height K (TSL_K)

- Union of theories

$$\begin{aligned}\mathsf{TSL}_\mathsf{K} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus T_{\mathsf{Reachability}} \oplus \\ & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level}_\mathsf{K}}\end{aligned}$$

$$\Sigma_{\mathsf{level}_\mathsf{K}} = \begin{array}{l} \{\mathsf{level}_\mathsf{K}\} \\ \left\{ \begin{array}{ccc} \eta_1 & : & \mathsf{level} \\ \vdots & & \\ \eta_\mathsf{K} & : & \mathsf{level} \end{array} \right\} \\ \emptyset \end{array}$$

Theory of Concurrent Skiplists of Height K ($\mathsf{TSL_K}$)

- Union of theories

$$
\begin{array}{rcl}
\mathsf{TSL_K} & = & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus T_{\mathsf{Reachability}} \oplus \\
 & & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level_K}}
\end{array}
$$

$$
\Sigma_{\mathsf{Bridge}} = \begin{array}{l}
\{\mathsf{mem}, \mathsf{addr}, \mathsf{set}, \mathsf{path}\} \\
\left\{ \begin{array}{lcl}
\mathit{path2set} & : & \mathsf{path} \rightarrow \mathsf{set} \\
\mathit{addr2set}_{\mathsf{K}} & : & \mathsf{mem} \times \mathsf{addr} \times \mathsf{level_K} \rightarrow \mathsf{set} \\
\mathit{getp}_{\mathsf{K}} & : & \mathsf{mem} \times \mathsf{addr} \times \mathsf{addr} \times \mathsf{level_K} \rightarrow \mathsf{path} \\
\mathit{firstlocked}_{\mathsf{K}} & : & \mathsf{mem} \times \mathsf{path} \times \mathsf{level_K} \rightarrow \mathsf{addr}
\end{array} \right\} \\
\left\{ \; \mathit{ordList} \; : \; \mathsf{mem} \times \mathsf{path} \; \right\}
\end{array}
$$

Theory of Concurrent Skiplists of Height K ($\mathsf{TSL_K}$)

- Union of theories

$$\begin{aligned}\mathsf{TSL_K} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus T_{\mathsf{Reachability}} \oplus \\ & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level_K}}\end{aligned}$$

Theory of Concurrent Skiplists of Height K (TSL_K)

- Union of theories

$$\begin{aligned}\mathsf{TSL}_\mathsf{K} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus T_{\mathsf{Reachability}} \oplus \\ & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level}_\mathsf{K}}\end{aligned}$$

- We want to use Nelson-Oppen

Theory of Concurrent Skiplists of Height K ($\mathsf{TSL_K}$)

- Union of theories

$$\begin{aligned}\mathsf{TSL_K} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus T_{\mathsf{Reachability}} \oplus \\ & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level_K}}\end{aligned}$$

- We want to use Nelson-Oppen
  - Stable infinite

Theory of Concurrent Skiplists of Height K ($\mathsf{TSL_K}$)

- Union of theories

$$
\begin{aligned}
\mathsf{TSL_K} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus T_{\mathsf{Reachability}} \oplus \\
& T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level_K}}
\end{aligned}
$$

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

Theory of Concurrent Skiplists of Height K ($\mathsf{TSL_K}$)

- Union of theories

$$\begin{aligned}\mathsf{TSL_K} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus T_{\mathsf{Reachability}} \oplus \\ & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level_K}}\end{aligned}$$

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

Theory of Concurrent Skiplists of Height K (TSL_K)

- Union of theories

$$\begin{array}{rcl} \mathsf{TSL}_\mathsf{K} & = & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus T_{\mathsf{Reachability}} \oplus \\ & & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level}_\mathsf{K}} \end{array}$$

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

Theory of Concurrent Skiplists of Height K ($\mathsf{TSL_K}$)

- Union of theories

$$\begin{aligned}\mathsf{TSL_K} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus T_{\mathsf{Reachability}} \oplus \\ & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level_K}}\end{aligned}$$

- 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

$$
\begin{aligned}
\mathsf{TSL}_\mathsf{K} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus T_{\mathsf{Reachability}} \oplus \\
& T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level}_\mathsf{K}}
\end{aligned}
$$

- 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 Σ-formula φ exists T-interpretation $\mathcal{A}$ satisfying φ s.t. $\mathcal{A}_\sigma$ 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 Σ-formula φ exists T-interpretation $\mathcal{A}$ satisfying φ s.t. $\mathcal{A}_\sigma$ is finite, for every $\sigma \in S_0$

Γ 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 Σ-formula φ exists T-interpretation $\mathcal{A}$ satisfying φ s.t. $\mathcal{A}_\sigma$ is finite, for every $\sigma \in S_0$

$$\Gamma \text{ a conjunction of } \mathsf{TSL}_\mathsf{K}\text{-literals}$$
$$\Downarrow$$
$$\text{a conjunction of normalized } \mathsf{TSL}_\mathsf{K}\text{-literals}$$

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 Σ-formula φ exists T-interpretation $\mathcal{A}$ satisfying φ s.t. $\mathcal{A}_\sigma$ is finite, for every $\sigma \in S_0$

Γ a conjunction of $\mathsf{TSL_K}$-literals

$$\Downarrow$$

a conjunction of normalized $\mathsf{TSL_K}$-literals

- Proof that exists a $\mathsf{TSL_K}$-interpretation $\mathcal{A}$
  - Bounded on K and Γ.
  - With finite number of elements in addr, elem, thid, ord and $\mathsf{level_K}$.

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 Σ-formula φ exists T-interpretation $\mathcal{A}$ satisfying φ s.t. $\mathcal{A}_\sigma$ is finite, for every $\sigma \in S_0$

Γ a conjunction of TSL_K-literals

$$\Downarrow$$

a conjunction of normalized TSL_K-literals

- Proof that exists a TSL_K-interpretation $\mathcal{A}$
  - Bounded on K and Γ.
  - With finite number of elements in addr, elem, thid, ord and $\mathsf{level}_\mathsf{K}$.

Γ is also T-satisfiable in $\mathcal{A}$

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 Σ-formula φ exists T-interpretation $\mathcal{A}$ satisfying φ s.t. $\mathcal{A}_\sigma$ is finite, for every $\sigma \in S_0$

Γ a conjunction of TSL_K-literals

$$\Downarrow$$

a conjunction of normalized TSL_K-literals

- Proof that exists a TSL_K-interpretation $\mathcal{A}$
  - Bounded on K and Γ.
  - With finite number of elements in addr, elem, thid, ord and $\mathsf{level}_\mathsf{K}$.

Γ is also T-satisfiable in $\mathcal{A}$

- TSL_K enjoys the small model property

A Decision Procedure for TSL_K

A Decision Procedure for TSL_K

$$\mathsf{TSL}_\mathsf{K} = \cdots \oplus T_{\mathsf{Reachability}} \oplus \cdots \oplus \textit{bridge functions and predicates}$$

A Decision Procedure for $\mathsf{TSL_K}$

$$\mathsf{TSL_K} = \cdots \oplus T_{\mathsf{Reachability}} \oplus \cdots \oplus \textit{bridge functions and predicates}$$

$$\begin{aligned} T_{\mathsf{Base}} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus T_{\mathsf{fseq}} \oplus \\ & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level_K}} \end{aligned}$$

A Decision Procedure for $\mathsf{TSL_K}$

$$\mathsf{TSL_K} = \cdots \oplus T_{\mathsf{Reachability}} \oplus \cdots \oplus \textit{bridge functions and predicates}$$

$$\begin{aligned} T_{\mathsf{Base}} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus \overbrace{T_{\mathsf{fseq}}}^{T_{\mathsf{path}}} \oplus \quad PATH \\ & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level_K}} \end{aligned}$$

A Decision Procedure for $\mathsf{TSL_K}$

$$\mathsf{TSL_K} = \cdots \oplus T_{\mathsf{Reachability}} \oplus \cdots \oplus \textit{bridge functions and predicates}$$

$$\begin{tikzcd}
\widehat{\mathsf{TSL_K}} & \\
 & T_{\mathsf{path}} \arrow[ul, squiggly, "GAP"'] \\
T_{\mathsf{Base}} \arrow[uu, squiggly] & T_{\mathsf{fseq}} \arrow[u, squiggly, "PATH"']
\end{tikzcd}$$

$$\begin{aligned}
T_{\mathsf{Base}} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus \overbrace{T_{\mathsf{fseq}}} \oplus \\
& T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level_K}}
\end{aligned}$$

A Decision Procedure for $\mathsf{TSL_K}$

$\mathsf{TSL_K} = \cdots \oplus T_{\mathsf{Reachability}} \oplus \cdots \oplus$ *bridge functions and predicates*

normalized $\mathsf{TSL_K}$-literals

$$\widehat{\mathsf{TSL_K}}$$

GAP

T_{path}

$PATH$

$$\begin{aligned} T_{\mathsf{Base}} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus \overbrace{T_{\mathsf{fseq}}} \oplus \\ & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level_K}} \end{aligned}$$

A Decision Procedure for $\mathsf{TSL_K}$

$$\mathsf{TSL_K} = \cdots \oplus T_{\mathsf{Reachability}} \oplus \cdots \oplus \textit{bridge functions and predicates}$$

normalized $\mathsf{TSL_K}$-literals

$$\widehat{\mathsf{TSL_K}}$$

GAP

$$T_{\mathsf{path}}$$

$PATH$

$$\begin{aligned} T_{\mathsf{Base}} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus \overbrace{T_{\mathsf{fseq}}} \oplus \\ & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level_K}} \end{aligned}$$

- By SMP it is always possible to enumerate all finitely many ground terms

A Decision Procedure for $\mathsf{TSL}_{\mathsf{K}}$

$$\mathsf{TSL}_{\mathsf{K}} = \cdots \oplus T_{\mathsf{Reachability}} \oplus \cdots \oplus \textit{bridge functions and predicates}$$

normalized $\mathsf{TSL}_{\mathsf{K}}$-literals

$$\widehat{\mathsf{TSL}_{\mathsf{K}}}$$

GAP

T_{path}

$PATH$

$$\begin{aligned} T_{\mathsf{Base}} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus \overbrace{T_{\mathsf{fseq}}} \oplus \\ & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level}_{\mathsf{K}}} \end{aligned}$$

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

A Decision Procedure for $\mathsf{TSL_K}$

$$\mathsf{TSL_K} = \cdots \oplus T_{\mathsf{Reachability}} \oplus \cdots \oplus \textit{bridge functions and predicates}$$

normalized $\mathsf{TSL_K}$-literals

$$\widehat{\mathsf{TSL_K}}$$

$$\widehat{\mathsf{TSL_K}} \xleftrightarrow{GAP} T_{\mathsf{path}} \xleftrightarrow{PATH} T_{\mathsf{fseq}}$$

$$\begin{aligned} T_{\mathsf{Base}} \quad = \quad & T_{\mathsf{addr}} \oplus T_{\mathsf{elem}} \oplus T_{\mathsf{cell}} \oplus T_{\mathsf{mem}} \oplus T_{\mathsf{fseq}} \oplus \\ & T_{\mathsf{set}} \oplus T_{\mathsf{setth}} \oplus T_{\mathsf{thid}} \oplus T_{\mathsf{mrgn}} \oplus T_{\mathsf{ord}} \oplus T_{\mathsf{level_K}} \end{aligned}$$

- By SMP it is always possible to enumerate all finitely many ground terms
- Unfolding of definitions in $PATH$ and 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

Small Model Property

Let Γ be a conjunction of normalized $\mathsf{TSL_K}$-literals. Let $\bar{e} = |V_{\mathsf{elem}}(\Gamma)|$, $\bar{a} = |V_{\mathsf{addr}}(\Gamma)|$, $\bar{m} = |V_{\mathsf{mem}}(\Gamma)|$, $\bar{p} = |V_{\mathsf{path}}(\Gamma)|$, $\bar{t} = |V_{\mathsf{thid}}(\Gamma)|$ and $\bar{o} = |V_{\mathsf{ord}}(\Gamma)|$. Then the following are equivalent:

- Γ is $\mathsf{TSL_K}$-satisfiable;
- Γ is true in a $\mathsf{TSL_K}$ interpretation $\mathcal{A}$ such that

$$
\begin{aligned}
&|\mathcal{A}_{\mathsf{addr}}| \leq \bar{a} + 1 + \bar{m}\,\bar{a} + \bar{p}^2 + \bar{p}^3 + (\mathsf{K}+2)\bar{m}\bar{p}\\
&|\mathcal{A}_{\mathsf{elem}}| \leq \bar{e} + \bar{m}\;|\mathcal{A}_{\mathsf{addr}}|\\
&|\mathcal{A}_{\mathsf{thid}}| \leq \bar{k} + \mathsf{K}\bar{m}\,|\mathcal{A}_{\mathsf{addr}}| + 1\\
&|\mathcal{A}_{\mathsf{level_K}}| \leq \mathsf{K}\\
&|\mathcal{A}_{\mathsf{ord}}| \leq \bar{o} + \bar{m}\;|\mathcal{A}_{\mathsf{addr}}|
\end{aligned}
$$

Small Model Property

$$
\begin{array}{lcl}
L & = & \{\eta_i \mid 1 \leq i \leq \mathsf{K}\} \\[2ex]
O & = & V_{\mathsf{ord}}^{\mathcal{B}} \cup \{m^{\mathcal{B}}(v).key^{\mathcal{B}} \mid m \in V_{\mathsf{mem}} \text{ and } v \in X\} \\[2ex]
X & = & V_{\mathsf{addr}}^{\mathcal{B}} \cup \{null^{\mathcal{B}}\} \cup \\
& & \{m^{\mathcal{B}}(v^{\mathcal{B}}).next^{\mathcal{B}} \mid m \in V_{\mathsf{mem}} \text{ and } v \in V_{\mathsf{addr}}\} \cup \\
& & \{v \in \delta(p^{\mathcal{B}}, q^{\mathcal{B}}) \mid \text{ the literal } p \neq q \text{ is in } \Gamma\} \cup \\
& & \{v \in \sigma({p_1}^{\mathcal{B}}, {p_2}^{\mathcal{B}}) \mid \text{ the literal } \neg append(p_1, p_2, p_3) \text{ is in } \Gamma \text{ and} \\
& & \qquad\qquad path2set^{\mathcal{B}}({p_1}^{\mathcal{B}}) \cap path2set^{\mathcal{B}}({p_2}^{\mathcal{B}}) \neq \emptyset\} \cup \\
& & \{v \in \sigma({p_1}^{\mathcal{B}} \circ {p_2}^{\mathcal{B}}, {p_3}^{\mathcal{B}}) \mid \text{ the literal } \neg append(p_1, p_2, p_3) \text{ is in } \Gamma \text{ and} \\
& & \qquad\qquad path2set^{\mathcal{B}}({p_1}^{\mathcal{B}}) \cap path2set^{\mathcal{B}}({p_2}^{\mathcal{B}}) = \emptyset\} \cup \\
& & \{v \in \kappa(m, p, l) \mid firstlocked(m, p, l) \text{ is in } \Gamma\} \\
& & \{v \in \xi(m, p) \mid \neg ordList(m, p) \text{ is in } \Gamma\} \\[2ex]
Y & = & V_{\mathsf{thid}}^{\mathcal{B}} \cup \{\oslash\} \cup \{m^{\mathcal{B}}(v).lockid^{\mathcal{B}} \mid m \in V_{\mathsf{mem}} \text{ and } v \in X\} \\[2ex]
Z & = & V_{\mathsf{elem}}^{\mathcal{B}} \cup \{m^{\mathcal{B}}(v).data^{\mathcal{B}} \mid m \in V_{\mathsf{mem}} \text{ and } v \in X\}
\end{array}
$$

$PATH$ definitions

$app : \mathsf{fseq} \times \mathsf{fseq} \to \mathsf{fseq}$
$app(nil, l) = l$ $app(cons(a, l), l') = cons(a, app(l, l'))$

$fseq2set : \mathsf{fseq} \to \mathsf{set}$
$fseq2set(nil) = \emptyset$ $fseq2set(cons(a, l)) = \{a\} \cup fseq2set(l)$

$ispath : \mathsf{fseq}$
$ispath(nil)$ $ispath(cons(a, nil))$ $\{a\} \nsubseteq fseq2set(l) \wedge ispath(l) \to ispath(cons(a, l))$

$last : \mathsf{fseq} \to \mathsf{addr}$
$last(cons(a, nil)) = a$ $l \neq nil \to last(cons(a, l)) = last(l)$

$isreachable : \mathsf{mem} \times \mathsf{addr} \times \mathsf{addr}$
$isreachable(m, a, a)$ $m[a].next = a' \wedge isreachable(m, a', b) \to isreachable(m, a, b)$

$isreachablep : \mathsf{mem} \times \mathsf{addr} \times \mathsf{addr} \times \mathsf{fseq}$
$isreachablep(m, a, a, nil)$ $m[a].next = a' \wedge isreachablep(m, a', b, p) \to isreachablep(m, a, b, cons(a, p))$

$firstmarked : \mathsf{mem} \times \mathsf{fseq} \times \mathsf{addr}$
$firstmarked(m, nil, null)$ $p \neq nil \wedge p = cons(j, q) \wedge m[j].lockid \neq \oslash \to firstmarked(m, p, j)$ $p \neq nil \wedge p = cons(j, q) \wedge m[j].lockid = \oslash \wedge firstmarked(m, q, i) \to firstmarked(m, p, i)$

GAP definitions

$$
\begin{array}{rcl}
nil & = & \epsilon \\
cons(a, nil) & = & [a] \\
\left(\begin{array}{c} ispath(p_1) \wedge ispath(p_2) \wedge \\ fseq2set(p_1) \cap fseq2set(p_2) = \emptyset \wedge \\ app(p_1, p_2) = p_3 \end{array} \right) & \leftrightarrow & append(p_1, p_2, p_3) \\
ispath\,(p) \rightarrow isreachable_{\mathsf{K}}\,(m, a, b, l, p) & = & reach_{\mathsf{K}}\,(m, a, b, l, p) \\
ispath\,(p) \rightarrow fseq2set(p) & = & path2set(p) \\
isreachable_{\mathsf{K}}\,(m, a, b, l, p) & \rightarrow & getp_{\mathsf{K}}\,(m, a, b, l) = p \\
\neg isreachable_{\mathsf{K}}\,(m, a, b, l, p) & \rightarrow & getp_{\mathsf{K}}\,(m, a, b, l) = nil \\
ispath\,(p) \wedge firstmarked\,(m, p, l, i) & \leftrightarrow & firstlocked\,(m, p, l) = i \\
ispath\,(p) \wedge ordPath\,(m, p) & \leftrightarrow & ordList\,(m, p)
\end{array}
$$