Chapter 5

Leaping reachability analysis

Thus far we have seen that FRA is a useful relief strategy for verifying multi-cyclic protocols. In particular, the execution of multiple transitions in one atomic step (viz. through fair transition-tuples) proved to be effective in reducing the number of global states and transitions examined for the purpose of deadlock detection in multi-cyclic protocols. Itoh & Ichikawa [II83] also employed this idea of executing multiple transitions concurrently for the verification of protocols with two or more processes in an arbitrary communication topology, but with restricted process structures (see Chapter 3). The idea of executing multiple transitions concurrently was ultimately generalized by Özdemir & Ural [ÖU94, ÖU95, Özd95], who proposed simultaneous reachability analysis (SRA) as a relief strategy for the verification of logical correctness properties of protocols with no topological or structural constraints at all. In essence, this generality is the result of allowing processes in a protocol to progress concurrently in a more flexible way than FRA.

In this chapter we propose an incremental improvement of SRA, called leaping reachability analysis (LRA), which maintains the power of SRA to detect all non-progress states, all non-executable transitions, all unspecified receptions and all buffer overflows in a protocol, while further reducing the size of the global state space that needs to be analyzed. This contribution was published incrementally in [SU96b, SU98a]. We start by formalizing LRA and then provide an analytical comparison between both relief strategies. An empirical comparison follows in Chapter 6.

5.1 Leap sets and proper leap sets

At the heart of LRA lies the concurrent execution of transitions at global states. Clearly, transitions that are to be executed concurrently must pertain to different processes. This foremost requirement is captured by the notion of leap sets in Definition 5.1.

Definition 5.1
Let $G$ be a global state of a protocol $\mathcal{P}$. A leap set in $G$ is a non-empty set $T$ of transitions
executable at \( G \) (i.e. \( T \subseteq X(G) \)) such that for all \( t, t' \in T \), \( \text{act}(t) \neq \text{act}(t') \) if \( t \neq t' \). The set of all leap sets in \( G \) is denoted by \( \text{leap}(G) \).

It follows that there are \(|T|!\) possible interleaving orders of the transitions in a leap set \( T \subseteq \text{leap}(G) \), all of which are equivalent up to \( \equiv \) and lead to the same global state when executed from \( G \) (cf. Definition 4.15 and Proposition 4.16).

**Definition 5.2**

Let \( G \) be a global state of a protocol \( \emptyset \) and \( T = \{ t_1, t_2, ..., t_k \} \subseteq \text{leap}(G) \). A linearization of \( T \) is a sequence \( t_{(1)} t_{(2)} ... t_{(k)} \), with \( \equiv \) any permutation on \( \{1, 2, ..., k\} \). The set of all linearizations of \( T \) is denoted by \( \text{lin}(T) \), and this notation is adopted also for sequences of leap sets: \( \text{lin}(T_1 T_2 ... T_m) = \{ \equiv_1 \equiv_2 ... \equiv_m \mid \equiv_i \in \text{lin}(T_i) \} \).

**Proposition 5.3**

Let \( T \subseteq \text{leap}(G) \) and \( \equiv \subseteq \text{lin}(T) \) with \( G \not\equiv_0^* H \), then \( T \equiv H \).

**Proof:** Directly from the fact that all transitions in \( T \) are executable at \( G \) and no two transitions in \( T \) belong to the same process.

Like the fair transition-tuples in FRA, leap sets provide a means to reduce the number of global states and transitions explored: all transitions in a leap set can in principle be executed concurrently in one atomic step. The reduction obtained increases with the size of a leap set and it is therefore tempting to consider only the maximal sets in \( \text{leap}(G) \). However, this is inadequate for verification purposes. To see this, suppose that some process \( P_i \) has a transition \( t \) that is potentially executable at \( G \) (see Section 4.1.2). Also assume that \( P_i \) has an executable transition \( t' \) at \( G \) that is included in some leap set \( T \) in \( G \). When all transitions in \( T \) are executed at once, \( t \) may become forever disabled by the execution of \( t' \) even though its possible execution at a global state \( H \) reachable from \( G \) could ultimately lead to a logical error. Figure 5.1 illustrates this scenario for a simple protocol. At the initial global state, the transitions \((10, -a, 11)\) and \((20, -b, 21)\) are executable and form the only maximal leap set. Transition \((20, +a, 22)\) is potentially executable and becomes executable at global state \((\emptyset 1, 20 | 22, 10)\), after the sole transmission of message \( a \). Its execution leads to the deadlock state \((\emptyset 1, 22 | 10, 1)\). Yet, when the two send transitions are executed concurrently, the reception of message \( a \) is no longer possible and this deadlock state is not detected.

In general, analyzing the effect of a potentially executable transition requires the corresponding process to refrain from progress for as long as the transition continues to be potentially executable [II83, ÖU94, ÖU95]. For instance, for the protocol above one must include a progress scheme which makes process \( P_2 \) wait until process \( P_1 \) has sent message \( a \). Based on this observation, we
employ a selective subset of \( \text{leap}(G) \) whose elements are called proper leap sets. Each proper leap set in \( G \) contains one executable transition from exactly those processes with executable transitions but no potentially executable transitions at \( G \), provided that some such process(es) exist(s). This ensures that the possible effect of potentially executable transitions at \( G \) is not ignored: processes with such transitions are forced to “wait” at their process states in \( G \) by excluding their transitions. The other processes are still forced to proceed concurrently in order to achieve state reduction. In the special case where each process with executable transitions at \( G \) also has at least one potentially executable transition at \( G \), there is little choice but to consider all executable transitions at \( G \) individually. Note that for each such transition \( t \) the singleton set \( \{ t \} \) is an element of \( \text{leap}(G) \).

**Definition 5.4**

Let \( G \) be a global state of a protocol \( G = (\{ P_i \mid i \in I \}, L) \). Define \( \text{wait}(G) = \{ i \} \mid X_i(G) \neq \emptyset, P_i(G) \neq \emptyset \} \), and the set \( \text{pleap}(G) \) of proper leap sets in \( G \) as follows:

\[
\text{pleap}(G) = \{ T \mid T \in \text{leap}(G) \land \text{act}(T) = \{ i \} \mid i \in \text{wait}(G) \}
\]

if \( \text{wait}(G) \neq \emptyset \)

\[
\text{pleap}(G) = \{ \{ t \} \mid t \in X(G) \}
\]

otherwise.

Note that trivially \( i \in \text{wait}(G) \) if no transition of process \( P_i \) is executable at \( G \). Some characteristic properties of \( \text{pleap}(G) \) are given in Proposition 5.5.

**Proposition 5.5**

i) \( t \in X_i(G) \land i \in \text{wait}(G) \land T \in \text{pleap}(G): t \in T \);

ii) \( T \in \text{pleap}(G) \land i \in \text{act}(T) \land t \in X_i(G) \land (T \setminus X_i(G)) \land \{ t \} \in \text{pleap}(G) \);

iii) \( T \in \text{pleap}(G) \land i \in \text{act}(T) \land \text{wait}(G) \land T \in \text{pleap}(G) \land \text{pleap}(G) = \bigcup_{\{ t \} \in X(G)} \{ t \} \).

**Proof:** Straightforward from Definition 5.4
The definition of *pleap*(G) is constructive and easily translated into an optimal algorithm. It first calculates the complement *I* \( \setminus \text{wait}(G) \) of *wait*(G) by inspecting the transition relations of all the processes. Clearly, this requires no overhead as the transition relations must be inspected equally for conventional reachability analysis. The algorithm then returns all (singleton sets of) transitions executable at G if *I* \( \setminus \text{wait}(G) = \emptyset \), or else the cross-product \( \prod_{i \notin \text{wait}(G)} X'_i(G) \). The overhead incurred by this step is \( O(|\prod_{i \notin \text{wait}(G)} X'_i(G)|) \).

**Example 5.6**
Consider the protocol \( \{P_1, P_2, P_3, P_4\} \), \( \{(1, 2), (2, 3), (3, 4), (4, 1), (4, 3)\} \), with the process graphs of processes \( P_1, P_2, P_3 \) and \( P_4 \) as in Figure 5.2 (by convention, \( m_{ij} \cap M_{ij} \)). Let
\[
\begin{align*}
t_1^1 &= (10, -m_{12}, 11) & t_1^2 &= (20, -m_{23}, 21) & t_1^3 &= (30, -m_{34}, 31) & t_1^4 &= (40, -m_{43}, 41) \\
t_2^1 &= (10, +m_{41}, 12) & t_2^2 &= (20, +m_{12}, 22) & t_2^3 &= (31, +m_{43}, 30) & t_2^4 &= (41, +m_{34}, 40)
\end{align*}
\]
then \( X(G^0) = \{ t_1^1, t_1^2, t_1^3, t_1^4 \} \) and \( P(G^0) = P^+(G^0) = \{ t_2^2, t_2^4 \} \). Hence, \( \text{leap}(G^0) = 2X(G^0) \setminus \{\emptyset\} \) and \( \text{wait}(G^0) = \{1, 2\} \), and thus \( \text{pleap}(G^0) = \{ t_1^1, t_1^2 \} \).

**Figure 5.2** A sample protocol.

### 5.2 Verifying indefinite progress

Following the discussion above, we formalize the execution of proper leap sets in global states and show that this is sufficient to detect all non-progress states of a protocol.

#### 5.2.1 \( \ell \)-reachability

**Definition 5.7**
Let \( G \) and \( H \) be global states of a protocol \( \prod_i G \uplus H \) iff \( \prod_T \text{pleap}(G) \) with \( \prod_{lin(T)} \) such that \( G \uplus H \). This is also denoted by \( G \uplus \uplus H \).

Well-definedness of \( \uplus \uplus \) follows from Proposition 5.3: executing a (proper) leap set in a global state \( G \) always yields a unique state \( H \).
Definition 5.8

Let \( G \) and \( H \) be global states of a protocol \( \mathcal{P} \), and denote by \( \varphi^* \) the reflexive and transitive closure of \( \varphi \). \( H \) is \( \ell \)-reachable from \( G \) iff \( G \varphi^* H \). When \( G = G^0 \), \( H \) is said to be \( \ell \)-reachable. The set of all \( \ell \)-reachable global states of \( \mathcal{P} \) is denoted by \( L_\mathcal{P} \). For a sequence of proper leap sets \( \mathbb{W} = T_1T_2\ldots T_m \), \( G \varphi_{\mathbb{W}}^* H \) denotes the existence of global states \( Q^0, Q^1, \ldots, Q^m \) such that \( G = Q^0 \varphi_{T_1} Q^1 \varphi_{T_2} \ldots \varphi_{T_m} Q^m = H \).

/* A is the set of global states that have been analyzed. */
/* W is the set of global states that still need to be analyzed. */

/* Initialize: */
A = \emptyset
W = \{G^0\}

/* Loop: */
while \( W \neq \emptyset \) do {
    remove an element \( G \) from \( W \)
    add \( G \) to \( A \)
    for all \( T \) in \text{pleap}(G) do {
        /* execution of proper leap set \( T \) */
        derive \( H \) such that \( G \varphi_{T} H \)
        if \( H \) is NOT already in \( A \) or \( W \) then add \( H \) to \( W \)
    }
}

Figure 5.3 State exploration by LRA.

An algorithm for exploring the \( \ell \)-reachable global state space, or \( \ell \)-reachability graph, of a protocol is shown in Figure 5.3. The box indicates the modification with respect to the standard perturbation algorithm in Figure 2.1. Clearly, every \( \ell \)-reachable global state is also reachable.

Proposition 5.9

\( L_\mathcal{P} \subseteq R_\mathcal{P} \)

Proof: By definition of \( \varphi^* \). 

5.2.2 Detecting non-progress states

The next lemma provides the basis for proving that the set \( L_\mathcal{P} \) of \( \ell \)-reachable global states includes all non-progress states and hence all deadlock states of a protocol.
**Lemma 5.10**

Let $G \not\equiv_{G,0}^* H$ and $\emptyset \neq \emptyset$ then there exist a proper leap set $T \not\subseteq \text{pleap}(G)$ with $\emptyset \not\subseteq \text{lin}(T)$, transition sequences $\emptyset$, $\emptyset$ and a global state $H$ such that $\emptyset \equiv_{H[\emptyset]}$, $\text{act}(\emptyset) \not\subseteq \text{act}(\emptyset) = \emptyset$ and $\emptyset \not\subseteq \emptyset$.

**Proof:** The proof essentially consists in showing that the diagram below holds true for some proper leap set $T \not\subseteq \text{pleap}(G)$, transition sequences $\emptyset$ and $\emptyset$ and global states $G$ and $H$. We show in particular that $T$, $\emptyset$ and $\emptyset$ can always be fixed such that $\emptyset$ contains exactly those transitions in $T$ that do not appear in $\emptyset$, and $\emptyset$ contains all transitions of $\emptyset$ except those that are in $T$.

$$
\begin{array}{c}
G \xrightarrow{T} G \\
\downarrow \hspace{2cm} \downarrow \hspace{2cm} \\
H \xrightarrow{\emptyset} H
\end{array}
$$

Since $\emptyset \neq \emptyset$, it follows that $X(G) \neq \emptyset$ and hence $\text{pleap}(G) \neq \emptyset$. Let $\text{first}(\emptyset) = \{ t \mid t \not\subseteq \emptyset \}$, $\text{pref}(\emptyset) \subseteq \{ i \not\subseteq \text{act}(\emptyset) \}$, i.e. $\text{first}(\emptyset)$ contains for each process active on $\emptyset$ the first transition of that process in $\emptyset$. Clearly, $\text{first}(\emptyset) \neq \emptyset$ since $\emptyset \neq \emptyset$. Now choose $T \not\subseteq \text{pleap}(G)$ such that $T \setminus \text{first}(\emptyset)$ is minimal ($\emptyset \not\subseteq T \setminus \text{first}(\emptyset)$). We show that $\text{act}(T \setminus \text{first}(\emptyset)) \not\subseteq \text{act}(\emptyset) = \emptyset$, by contradiction. Suppose $i \not\subseteq T \setminus \text{first}(\emptyset)$, then $i \not\subseteq \text{act}(T \setminus \text{first}(\emptyset))$ and let $t$, $t \subseteq \emptyset$, such that $t \not\subseteq T \setminus \text{first}(\emptyset)$ and $t \subseteq \text{first}(\emptyset)$, then $i \not\subseteq X_i(G) \not\subseteq P_i(G)$. By Proposition 5.5.(ii), if $i \not\subseteq X_i(G)$ then $(T \setminus \{t\}) \not\subseteq \{ i \} \not\subseteq \text{pleap}(G)$. However, $|(T \setminus \{t\}) \not\subseteq \{ i \} \setminus \text{first}(\emptyset)| < |T \setminus \text{first}(\emptyset)|$ which contradicts the minimality of $T \setminus \text{first}(\emptyset)$.

On the other hand, if $i \not\subseteq P_i(G)$ then $i \not\subseteq \text{act}(T) \not\subseteq \text{wait}(G)$ and thus $\text{pleap}(G) = \bigcup_{G \in X(G)} \{t\}$, by Proposition 5.5.(iii), which in turn implies that $T = T \setminus \text{first}(\emptyset) = \{t\}$. In this case let $t''$ be the very first transition of $\emptyset$, then $t'' \not\subseteq X(G)$, $\{ t'' \} \not\subseteq \text{pleap}(G)$ and moreover $\{ t'' \} \not\subseteq \text{first}(\emptyset)$. Again, this contradicts the minimality of $T \setminus \text{first}(\emptyset)$, since $\{ t'' \} \setminus \text{first}(\emptyset) = \emptyset \not\subseteq T \setminus \text{first}(\emptyset)$. Hence, the claim $\text{act}(T \setminus \text{first}(\emptyset)) \not\subseteq \text{act}(\emptyset) = \emptyset$ holds.

Let $\emptyset \subseteq \text{lin}(T \setminus \text{first}(\emptyset))$ if $T \setminus \text{first}(\emptyset) \neq \emptyset$, or else $\emptyset = \emptyset$. Surely, $\text{act}(\emptyset) \not\subseteq \text{act}(\emptyset) = \emptyset$ in either case and thus there exists a global state $H$ such that $G \not\equiv_{G,0}^* H$(by Corollary 4.4). Now, let $\emptyset \subseteq \text{lin}(T)$ and such that $\emptyset \equiv_{H[\emptyset]}$ with $\emptyset \subseteq \text{lin}(T \setminus \text{first}(\emptyset))$. As $\emptyset \equiv_{H[\emptyset]}$ and $\text{act}(\emptyset) \not\subseteq \text{act}(\emptyset) = \emptyset$, it follows that also $G \not\equiv_{G,0}^* H$. Thus, $\emptyset \not\subseteq \text{lin}(T \setminus \text{first}(\emptyset))$, $\text{act}(\emptyset) \not\subseteq \text{act}(\emptyset) = \emptyset$ and $\emptyset \not\subseteq \emptyset$, which proves the lemma.

When $H$ is a non-progress state, $\emptyset$ is empty and the conclusion of Lemma 5.10 reduces to $\emptyset \equiv_{H[\emptyset]}$ and thus $\emptyset \neq \emptyset$. By repeated application of this lemma one can then prove that $H$ is $\ell$-reachable.

**Theorem 5.11**

Every non-progress state is $\ell$-reachable.
**Proof:** Let $H \not\subseteq R_{\emptyset}$ be a non-progress state and $G^0 \not\subseteq H$. $H$ is trivially $\ell$-reachable if $|\emptyset| = 0$ since then $H = G^0$. Suppose that $|\emptyset| > 0$. By Lemma 5.10 there is a proper leap set $T_1 \subseteq \text{pleap}(G^0)$ with $\emptyset \subseteq \text{lin}(T_1)$, transition sequences $\emptyset_1$ and $\emptyset_1$, and a global state $H^1$ such that $\emptyset \subseteq \emptyset_1 \subseteq \emptyset_1$ and $|\emptyset_1| \not\subseteq |\emptyset|$. Let $G^0 \not\subseteq G^1 \not\subseteq H^1$, then $|\emptyset_1| > 0$ equally implies the existence of $T_2 \subseteq \text{pleap}(G^1)$ with $\emptyset_2 \subseteq \text{lin}(T_2)$, $\emptyset_2$, and $H^2$ such that $\emptyset \subseteq \emptyset_2 \subseteq \emptyset_2$ and $|\emptyset_2| \not\subseteq |\emptyset_1|$. Applying this argument repeatedly results in the following diagram:

$$
\begin{array}{cccccc}
G^0 & \xrightarrow{T_1} & G^1 & \xrightarrow{T_2} & G^2 & \cdots & G^m \\
\emptyset & \emptyset_1 & \emptyset_2 & \cdots & H^m \\
H & H^1 & H^2 & \cdots & \emptyset
\end{array}
$$

Thus $\emptyset_{j-1} \emptyset_j \emptyset \not\subseteq \emptyset$ and $|\emptyset_j| \not\subseteq |\emptyset_{j-1}|$, with $\emptyset \subseteq \text{lin}(T_j)$, $T_j \subseteq \text{pleap}(G^{j-1})$ and $|\emptyset_j| > 0$ ($2 \leq j \leq m$). As a result, $\emptyset \emptyset_1 \emptyset_2 \cdots \emptyset_m \subseteq \emptyset$ and $H^m = H$, implying that $|\emptyset_j| < |\emptyset_{j-1}|$, for all $2 \leq j \leq m$, and $|\emptyset_1| < |\emptyset|$. We may then assume that $|\emptyset_m| = 0$ for $\emptyset$ is finite and Lemma 5.10 applies as long as $|\emptyset_j| > 0$. Thus, $\emptyset \subseteq H^m \subseteq \emptyset_1 \emptyset_2 \cdots \emptyset_m$ and $G^0 \not\subseteq H^m \emptyset \subseteq H$, i.e. $H$ is $\ell$-reachable.

**Corollary 5.12**

Every deadlock state is $\ell$-reachable.

**Corollary 5.13**

Indefinite progress and deadlock-freedom for a protocol $\emptyset$ are decidable if $L_{\emptyset}$ is finite.

![Figure 5.4](image)

**Figure 5.4** The $\ell$-reachability graph of the protocol of Example 5.6.

**Example 5.14**

The $\ell$-reachability graph of the protocol of Example 5.6 (see Figure 5.2) is shown in Figure 5.4. For the initial global state we have $\text{pleap}(G^0) = \{ \{t^1_3, t^4_3\} \}$. Executing proper leap set $\{t^1_3, t^4_3\}$ in $G^0$ results in the global state $\emptyset \emptyset 0, 20, 31, 41 \emptyset m_{34}, m_{43} \emptyset$. This state has also one proper leap set, namely $\{t^2_3, t^2_4\}$, and its execution leads back to the initial global state. State exploration
based on \( \mathcal{G}^* \) thus terminates, and since both states are progress states the protocol is (correctly) found to progress indefinitely. Of course, the same result is obtained by conventional reachability analysis but this requires exploring as many as 40 global states and 100 global state transitions.

While undecidability prevails in general, it is again interesting to note that LRA, like FRA (cf. Section 4.3.2), is effective beyond the class of bounded protocols. For instance, consider a small modification of the protocol of Example 5.6, replacing transition \((10, \neg m_{12}, 11)\) of process \(P_1\) by the “cyclic” transition \((10, \neg m_{12}, 10)\). It is easy to see that the resulting protocol is unbounded, whereas its \(\ell\)-reachability graph is the same as in Figure 5.4 and thus finite. Another interesting result is the fact that the relation \( \mathcal{G}^* \) preserves the possibility of indefinite progress of a protocol, in the sense stated by Proposition 5.15.

**Proposition 5.15**
There exists an infinite sequence of reachable global states in the (conventional) reachability graph of a protocol \(\perp\) iff there exists an infinite sequence of \(\ell\)-reachable global states in the \(\ell\)-reachability graph of \(\perp\).

**Proof:** The “if” part is immediate by definition of the \(\ell\)-reachability relation \( \mathcal{G}^* \). For the “only-if” part, assume the existence of an infinite sequence of reachable global states in the reachability graph of \(\perp\), with \(\perp\) the corresponding infinite transition sequence from the initial global state \(G^0\).

Let \(\perp = \{\cdot\}\) such that \(\text{act}(\perp) = \text{act}(\perp)\), i.e. \(\{\cdot\}\) \(\text{pref}(\perp)\) contains at least one transition from each process that is active on \(\perp\). We have \(\perp \neq \{\cdot\} G^0 \xrightarrow{\ell} H\) and \(\perp\) is an infinite transition sequence from \(H\), for some reachable global state \(H\). By Lemma 5.10, there exist \(T \xrightarrow{\perp} \text{pleap}(G^0)\) with \(\{\cdot\} \xrightarrow{\perp} \text{lin}(T)\), transition sequences \(\{\cdot\}\) and \(\{\cdot\}\) and a global state \(H\) such that \(\{\cdot\} G^0 \xrightarrow{\ell} \{\cdot\}, \text{act}(\{\cdot\}) = \emptyset\) and \(\{\cdot\}\) \(\text{pref}(\perp)\) contains at least one transition from each process that is active on \(\perp\). By the choice of \(\{\cdot\}\), \(\text{act}(\{\cdot\}) = \emptyset\) and thus \(\text{act}(\{\cdot\}) = \emptyset\). It follows that \(\{\cdot\}\) is also an infinite transition sequence from \(H\)

\[
\begin{array}{ccc}
G^0 & \xrightarrow{\ell} & G \\
\downarrow & & \downarrow \\
\{\cdot\} & \xrightarrow{\perp} & H \\
\downarrow & & \downarrow \\
\{\cdot\} & \xrightarrow{\perp} & \{\cdot\}
\end{array}
\]

Since \(\{\cdot\}\) is infinite we can apply the same reasoning infinitely often, continuing with the sequence \(\{\cdot\}\) from \(G\) to yield an infinite sequence of \(\ell\)-reachable global states.

\[\square\]
5.3 Verifying freedom of non-executable transitions

State exploration by LRA based on the relation $\varphi^*$ is inadequate for verifying the absence of non-executable transitions. This is witnessed by the protocol of Example 5.6: transition $(20, +m_{12}, 22)$ of process $P_2$ is surely executable, but not at an $\ell$-reachable global state (see Figure 5.4). Thus, a decision procedure based on $\varphi^*$ would mistakenly report this transition as being non-executable. The problem lies in the conception of delaying the processes with potentially executable transitions at global states, which results in a phenomenon referred to as the ignoring problem [Val90]: some processes may be delayed indefinitely from a certain global state, causing the behavior of these processes to be ignored. For instance, the behavior of processes $P_1$ and $P_2$ of the protocol of Example 5.6 is completely ignored because in each $\ell$-reachable global state they have potentially executable transitions, while $P_3$ and $P_4$ always have exclusively executable transitions.

In general, due to the ignoring problem the relation $\varphi^*$ may not expose all reachable process states (i.e. process states appearing in some reachable global state). This clearly hinders not only the detection of non-executable transitions, but that of unspecified receptions and buffer overflows as well. For the verification of logical correctness properties other than indefinite progress and deadlock-freedom we must therefore augment the state exploration scheme defined in the previous section. Preferably, it should be adapted to the property one wants to verify. This section presents a simple extension of the $\ell$-reachability relation $\varphi^*$ for detecting the non-executable transitions of a protocol. Unspecified receptions and buffer overflows are dealt with in Section 5.4.

5.3.1 $\ell^*$-reachability

An effective solution to the ignoring problem is obtained by executing a minimal number of extra leap sets in a given global state $G$, in addition to the proper leap sets, such that each executable transition at $G$ is executed via at least one leap set. Precisely, we extend $pleap(G)$ to form the set $xpleap(G)$ by adding for one arbitrary $T \subseteq pleap(G)$ all the leap sets $T \not\subseteq \{t\}$, where $t$ is a transition executable at $G$ but not included in any proper leap set in $pleap(G)$. This extension ensures that processes with executable transitions are not expelled from progress.

**Definition 5.16**

Let $G$ be a global state of a protocol $\square$ and $T \subseteq pleap(G)$ an arbitrary proper leap set in $G$. Define

$$xpleap(G) = pleap(G) \cup \{T \not\subseteq \{t\} \mid t \not\subseteq X(G) \not\subseteq act(t) \not\subseteq wait(G)\}$$

if $wait(G) \not\subseteq I$.

$$xpleap(G) = pleap(G)$$

otherwise.

□
Note that $xpleap(G) = pleap(G)$ also when $wait(G) = \emptyset$. Further note that if $pleap(G)$ consists of more than one proper leap set without capturing all executable transitions at $G$, then $xpleap(G)$ is not unique since the proper leap set to be used for extension can be selected non-deterministically. Yet, we stress that this nondeterminism does not affect in any way the theoretical results in this section. For the mere sake of convenience, we use as a heuristic to always choose the first element of $pleap(G)$ when viewed as an ordered set. This allows us to refer to $xpleap(G)$ as a unique set and an algorithm for its construction then follows directly from Definition 5.16. Clearly, the fact that we need to fix only one proper leap set for extension (as opposed to all of them) is important for efficiency considerations, particularly when $|pleap(G)|$ is large.

Insightful properties of the set $xpleap(G)$ are stated in Proposition 5.17, analogous to those of $pleap(G)$ in Proposition 5.5. Observe especially that $\bigcup_{T \in xpleap(G)} T = X(G)$, i.e. for each executable transition $t$ at $G$ there is a leap set in $xpleap(G)$ containing $t$ (Proposition 5.17.(i)), which does not necessarily hold for $pleap(G)$ (cf. Proposition 5.5.(i)).

**Proposition 5.17**

- $i) \quad t \not\in X(G) \implies T \not\in xpleap(G); \; t \in T$;
- $ii) \quad T \not\in xpleap(G) \implies \exists i \in act(T) \not\in wait(G) \not\in \bigcup_{T \in X_i(G)} \{ t \} \not\in xpleap(G)$;
- $iii) \quad T \not\in xpleap(G) \exists i \in act(T) \not\in wait(G) \not\in \bigcup_{T \in X_i(G)} \{ t \} \not\in xpleap(G)$.

**Proof:** Straightforward from Definition 5.16.

**Example 5.18**

Consider once again the protocol of Example 5.6: $X(G^0) = \{ t_1^1, t_1^2, t_3^1, t_4^1 \}$, $wait(G^0) = \{ 1, 2 \}$ and $pleap(G^0) = \{ \{ t_1^1 \}, \{ t_2^1 \} \}$. That is, all four processes have executable transitions at $G^0$, but process $P_1$ and process $P_2$ also have potentially executable transitions at $G^0$. With neither transition $t_1^1$ nor transition $t_2^1$ occurring in a proper leap set in $G^0$, by the first clause of Definition 5.16 we have $xpleap(G^0) = \{ \{ t_3^1, t_3^2 \}, \{ t_3^1, t_4^1, t_4^1 \}, \{ t_3^1, t_4^1, t_2^1 \} \}$.

As before, we define a reachability relation among global states which governs the execution of all elements of $xpleap(G)$.

**Definition 5.19**

Let $G$ and $H$ be global states of a protocol $\mathcal{G}$, $G \not\approx H$ iff $\bigcup T \not\in \bigcup X(T)$ of $xpleap(G)$ with $\bigcup \bigcup lin(T)$ such that $G \not\approx_0 H$. This is also denoted by $G \not\approx H$.

**Definition 5.20**

Let $G$ and $H$ be global states of a $\mathcal{G}$, and denote by $\not\approx_0^*$ the reflexive and transitive closure of $\not\approx_0$. 
$H$ is $\ell^*$-reachable from $G$ iff $G \subseteq H$. If $G = G^0$, then $H$ is said to be $\ell^*$-reachable. The set of all $\ell^*$-reachable global states of $\mathcal{G}$ is denoted by $L^*_\mathcal{G}$. For a sequence of leap sets $\mathcal{G} = T_1T_2\ldots T_m$, $G \subseteq^* H$ denotes the existence of global states $Q^0, Q^1, \ldots, Q^m$ such that $G = Q^0 \subseteq^* Q^1 \subseteq^* \ldots \subseteq^* Q^m = H$.

Surely, an algorithm for constructing the $\ell^*$-reachability graph of a protocol (representing its $\ell^*$-reachable global state space) is akin to the one in Figure 5.3. We arrive at some anticipated results. By definition of the relation $\subseteq^* \mathcal{G}$ as an extension of $\subseteq^* \mathcal{G}$ it follows that the $\ell$-reachability graph is a subgraph of the $\ell^*$-reachability graph. Consequently, all $\ell$-reachable global states and thus all non-progress states of a protocol (by Theorem 5.11) are $\ell^*$-reachable.

**Proposition 5.21**

$L^*_\mathcal{G} \subseteq R^*_\mathcal{G}$

**Proof:** By definition of xpleap($G$) we have pleap($G$) $\subseteq$ xpleap($G$), for any global state $G$. This implies $L^*_\mathcal{G} \subseteq R^*_\mathcal{G}$. The inclusion $L^*_\mathcal{G} \subseteq R^*_\mathcal{G}$ holds by definition of $\subseteq^* \mathcal{G}$.

**Corollary 5.22**

Every non-progress state is $\ell^*$-reachable.

**Corollary 5.23**

Every deadlock state is $\ell^*$-reachable.

### 5.3.2 Detecting non-executable transitions

The relation $\subseteq^* \mathcal{G}$ is intended for detecting non-executable transitions. We prove that exploring the $\ell^*$-reachability graph is indeed sufficient for this purpose: for each executable transition of a protocol $\mathcal{G}$, there is at least one global state in $L^*_\mathcal{G}$ at which the transition is executable. As a result, LRA based on $\subseteq^* \mathcal{G}$ can verify the absence of non-executable transitions (and non-progress states) for any protocol with a finite $\ell^*$-reachability graph.

**Lemma 5.24**

Let $G \subseteq^* H$ and $\mathcal{G} \neq \mathcal{G}$ then there exist $T \mathcal{G}$ xpleap($G$) with $\mathcal{G}$ $\subseteq$ lin($T$), transition sequences $\mathcal{G}$, $\mathcal{G}$ and a global state $H\mathcal{G}$ such that $\mathcal{G} G \subseteq H\mathcal{G}$, act($\mathcal{G}$) $\subseteq$ act($\mathcal{G}$) = $\varnothing$ and $|\mathcal{G}| < |\mathcal{G}|$.

**Proof:** By Lemma 5.10, we have a proper leap set $T \mathcal{G}$ xpleap($G$) $\subseteq$ xpleap($G$) with $\mathcal{G}$ $\subseteq$ lin($T \mathcal{G}$), transition sequences $\mathcal{G}$ and $\mathcal{G}$ and a global state $H\mathcal{G}$ such that $\mathcal{G} G \subseteq H\mathcal{G}$, act($\mathcal{G}$) $\subseteq$ act($\mathcal{G}$) = $\varnothing$ and $|\mathcal{G}| = |\mathcal{G}|$. We know from the proof of Lemma 5.10 that $T \mathcal{G}$ and $\mathcal{G}$ are such that $T \mathcal{G}$ first($\mathcal{G}$) is
minimal and \[\mathcal{L}(T \backslash \text{first}(\square))\] where \(\text{first}(\square) = \{t^r \mid t \in G, \text{pref}(t) \subseteq i \text{ act}(\square)\}.\) Clearly, if \(T \backslash \text{first}(\square) \neq T\) then \(|\square| < |\square|\) and hence \(|\square| < |\square|\). Thus, in this case the lemma holds, i.e. choose \(T = T \backslash \square = \emptyset\) and \(T = \emptyset\).

Alternatively, if \(T \backslash \text{first}(\square) = T\) then \(\square = \mathcal{L}(\square)\) and \(\text{act}(\square) \subseteq \text{act}(T) = \emptyset.\) We must have \(\text{wait}(G) \subseteq I\) because otherwise \(\text{pleap}(G) = \bigcup_{t \in \text{X}(G) \setminus \{t\}}\{t\} \setminus \text{first}(\square) = 0 < |T \backslash \text{first}(\square)|,\) with \(t\) the first transition of \(\square\), which contradicts the minimality of \(T \backslash \text{first}(\square)\). Definition 5.4 then states that \(\text{act}(T) = \{i \mid i \in \text{wait}(G)\}\) for all \(T \subseteq \text{pleap}(G)\). Since \(\text{act}(\square) \subseteq \text{act}(T) = \emptyset\) it follows that \(\text{act}(\square) \subseteq \text{wait}(G)\) and \(T \subseteq \text{pleap}(G)\): \(\text{act}(\square) \subseteq \text{act}(T) = \emptyset.\) In particular, the first transition \(t\) of \(\square\) is executable at \(G\) but not in any proper leap set in \(G\). According to Definition 5.16, in this case \(\text{pleap}(G) \subseteq \text{xpleap}(G)\) and some proper leap set in \(G,\) say \(T'' \subseteq \text{pleap}(G)\), is selected to form \(\text{xpleap}(G)\). Now choose \(T = T'' \subseteq \{t\}\), then \(T \subseteq \text{xpleap}(G), T \backslash \text{first}(\square) = T'' \backslash \text{first}(\square) = T''\) and \(\text{act}(T) \subseteq \text{act}(\square) = \text{act}(t)\). Let \(\square \subseteq \text{lin}(T), \square \subseteq \text{lin}(T'')\) and \(\square = t\). Then \(\subseteq \text{act}(\square) = \emptyset\) and \(|\square| < |\square|\), i.e. again the lemma holds.

Notice that Lemma 5.24 differs from Lemma 5.10 in the strict inequality \(|\square| < |\square|\), which now holds irrespective of whether \(H\) is a non-progress state. This enables the following generalization of Lemma 5.24 to sequences of leap sets.

**Lemma 5.25**

Let \(G \not\equiv_{\square}^r H\) and \(\square \neq \emptyset\) then there exist a sequence of leap sets \(\square\) with \(\emptyset \subseteq \text{lin}(\square)\), a transition sequence \(\emptyset\) and a global state \(H\) such that \(G \not\equiv_{\square}^r H\) and \(\emptyset \subseteq \text{lin}(\square)\).

**Proof:** The proof essentially consists in showing that the following diagram holds true for some sequence of leap sets \(\square = T_1 T_2 \ldots T_m\), transition sequence \(\emptyset\) and a global state \(H\).

According to Lemma 5.24, there exist \(T_1 \subseteq \text{xpleap}(G)\) with \(\emptyset \subseteq \text{lin}(T_1)\), \(\emptyset_1\) and \(H_1\) such that \(\emptyset \subseteq \emptyset_1 \subseteq H_1\) and \(|\emptyset| < |\emptyset|\). Let \(G \not\equiv_{\square}^r G^1 \not\equiv_{\square}^r H^1\) then \(|\emptyset| > 0\) equally implies the existence of \(T_2 \subseteq \text{xpleap}(G^1)\) with \(\emptyset_2 \subseteq \text{lin}(T_2)\), \(\emptyset_2\) and \(H^2\) such that \(\emptyset \subseteq \emptyset_2 \subseteq H^2\) and \(|\emptyset_2| < |\emptyset|\). As in the proof of Theorem 5.11, applying this argument \(m\) times yields \(\emptyset \subseteq \emptyset_1 \subseteq \emptyset_2 \subseteq \ldots \subseteq \emptyset_m \subseteq \ldots \subseteq \emptyset_m\). Since \(\emptyset\) is finite and Lemma 5.24 can be applied as long as \(|\emptyset| > 0\) (\(1 \not\sqsubseteq j \not\sqsubseteq m\)), we may assume that \(|\emptyset_m| = 0\), i.e. \(\emptyset_1 \subseteq \ldots \subseteq \emptyset_m \subseteq \emptyset_1 \subseteq \ldots \subseteq \emptyset_m\). Let \(\emptyset = T_1 T_2 \ldots T_m\), \(\emptyset = \emptyset_1 \subseteq \ldots \subseteq \emptyset_m \subseteq \text{lin}(\emptyset)\), \(\emptyset \subseteq \emptyset_1 \subseteq \ldots \subseteq \emptyset_m\) and \(H \subseteq H^m\), then \(G \not\equiv_{\square}^r H\) and \(\emptyset \subseteq \text{lin}(\emptyset)\).
Theorem 5.26
A transition \( t \) is executable iff \( t \) is executable at an \( \ell^* \)-reachable global state.

Proof: The “if” part holds directly since \( L_{\| G}^* \sqsubseteq R_G \). For the “only-if” part, when \( t \) is executable there must be a reachable global state \( H \) such that \( G^0 \stackrel{t\| H}{\rightarrow}^* H \), for some transition sequence \( \| \). By Lemma 5.25, there exist \( \| \) with \( \| \) \( \text{lin}(\|) \), \( \| \) and \( H \) such that \( G^0 \stackrel{t\| H}{\rightarrow}^* H \) and \( \| \) \( G^0 \| \). That is, \( t \) appears in \( \| \) and is thus executable at an \( \ell^* \)-reachable global state. \( \square \)

Corollary 5.27
Freedom of non-executable transitions for a protocol \( \| \) is decidable if \( L_{\| G}^* \) is finite.

\[ G^0 = \{ 0, 20, 30, 40 \| \} \]

\[ G^1 = \{ 1, 20, 31, 41 \| \{ m_{12}, m_{43} \} \} \]

\[ G^2 = \{ 0, 20, 30, 40 \| \{ m_{12}, m_{43} \} \} \]

\[ G^3 = \{ 0, 21, 31, 41 \| \{ m_{23}, m_{34}, m_{43} \} \} \]

\[ G^4 = \{ 1, 20, 31, 41 \| \{ m_{12} \} \} \]

\[ G^5 = \{ 0, 21, 30, 40 \| \{ m_{23} \} \} \]

\[ G^6 = \{ 1, 22, 30, 40 \| \} \]

\[ G^7 = \{ 1, 21, 31, 41 \| \{ m_{12}, m_{23}, m_{34}, m_{43} \} \} \]

\[ G^8 = \{ 1, 22, 31, 41 \| \{ m_{34}, m_{43} \} \} \]

\[ G^9 = \{ 1, 21, 30, 40 \| \{ m_{12}, m_{23} \} \} \]

**Figure 5.5** The \( \ell^* \)-reachability graph of the protocol of Example 5.6.

Example 5.28
Figure 5.5 shows the \( \ell^* \)-reachability graph of the protocol of Example 5.6, consisting of 10 \( \ell^* \)-reachable global states and 18 global state transitions (empty channels are not indicated). For each \( \ell^* \)-reachable global state \( G^k \), Table 5.1 lists the data used to calculate the set \( \text{xpleap}(G^k) \) of leap sets executed at \( G^k \). The elements of \( \text{xpleap}(G^k) \) are precisely the labels of the outgoing edges of the
node labeled \( G^k \). Again the protocol is found to progress indefinitely (all the nodes in the \( \ell^e \)-reachability graph have outgoing edges) and, moreover, state exploration by LRA based on \( \varnothing G^0 \) correctly reveals one non-executable transition, namely \( t_1^2 = (10, +m_{12}, 12) \) \( t_1^2 \) does not occur in the \( \ell^e \)-reachability graph). Recall that a notably larger number of global states and transitions are explored in conventional reachability analysis (40 and 100, respectively).

\[ \square \]

<table>
<thead>
<tr>
<th>Global state ( G^k )</th>
<th>( X(G^k) )</th>
<th>( P(G^k) )</th>
<th>( \text{wait}(G^k) )</th>
<th>( \text{pleap}(G^k) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G^0 )</td>
<td>( {t_1^1, t_2^1, t_3^1, t_4^1} )</td>
<td>( {t_1^2, t_2^2} )</td>
<td>( {1, 2} )</td>
<td>( {(t_1^1, t_4^1)} )</td>
</tr>
<tr>
<td>( G^1 )</td>
<td>( {t_1^2, t_2^2, t_3^2, t_4^2} )</td>
<td>( \varnothing )</td>
<td>( {1} )</td>
<td>( {(t_1^2, t_3^2), (t_2^2, t_3^2, t_4^2)} )</td>
</tr>
<tr>
<td>( G^2 )</td>
<td>( {t_1^1, t_2^1, t_3^1, t_4^1} )</td>
<td>( {t_1^2, t_2^2} )</td>
<td>( {1, 2} )</td>
<td>( {(t_2^2, t_3^2)} )</td>
</tr>
<tr>
<td>( G^3 )</td>
<td>( {t_1^1, t_2^1, t_3^1, t_4^1} )</td>
<td>( \varnothing )</td>
<td>( {1} )</td>
<td>( {(t_1^1, t_3^1, t_4^1), (t_2^2, t_3^2, t_4^2)} )</td>
</tr>
<tr>
<td>( G^4 )</td>
<td>( {t_1^1, t_4^1} )</td>
<td>( {t_1^2} )</td>
<td>( {1, 2} )</td>
<td>( {(t_1^1, t_4^1)} )</td>
</tr>
<tr>
<td>( G^5 )</td>
<td>( {t_3^1, t_4^1} )</td>
<td>( \varnothing )</td>
<td>( {1, 2} )</td>
<td>( {(t_1^1, t_4^1)} )</td>
</tr>
<tr>
<td>( G^6 )</td>
<td>( {t_3^2, t_4^2} )</td>
<td>( \varnothing )</td>
<td>( {1, 2} )</td>
<td>( {(t_3^2, t_4^2)} )</td>
</tr>
<tr>
<td>( G^7 )</td>
<td>( {t_3^2, t_4^2} )</td>
<td>( \varnothing )</td>
<td>( {1, 2} )</td>
<td>( {(t_3^2, t_4^2)} )</td>
</tr>
<tr>
<td>( G^8 )</td>
<td>( {t_3^2, t_4^2} )</td>
<td>( \varnothing )</td>
<td>( {1, 2} )</td>
<td>( {(t_3^2, t_4^2)} )</td>
</tr>
</tbody>
</table>

As pointed out earlier, the \( \ell \)-reachable global state space \( \mathbf{L}_U \) may be finite even for unbounded protocols. The same is true for \( \mathbf{L}_U^* \) as witnessed by the simple protocol with two processes \( P_1 \) and \( P_2 \), where \( P_1 \) consists of the single cyclic transition \( (10, -m_{12}, 10) \) and \( P_2 \) the single cyclic transition \( (20, +m_{12}, 20) \). Assuming that the simplex channel from \( P_1 \) to \( P_2 \) is not prebounded, this protocol has an infinite number of reachable global states but only two \( \ell^e \)-reachable global states. Yet, the class of unbounded protocols with finite \( \mathbf{L}_U^* \) is properly included in the class of unbounded protocols with finite \( \mathbf{L}_U \). This can be seen from the protocol of Example 5.6 with the cyclic transition \( (10, -m_{12}, 10) \) replacing transition \( (10, -m_{12}, 11) \) of process \( P_1 \). Recall from the discussion following Example 5.14 that the \( \ell \)-reachability graph of this protocol is finite (see Figure 5.4). One may check that its \( \ell^e \)-reachability graph is infinite.

### 5.4 Verifying freedom of unspecified receptions & buffer overflows

Despite its qualification to solve the ignoring problem and thereby the problem of detecting non-executable transitions, the \( \ell^e \)-reachability relation is still unsuited for the detection of unspecified receptions (ur-pairs) and buffer overflows (bo-pairs). The protocol of Example 5.6 serves again as
an example. Two ur-pairs \((30, m_{43})\) and \((40, m_{34})\) occur in, for instance, the reachable global states \((\emptyset, 0, 20, 30, 41, \emptyset, \emptyset, \emptyset, m_{34}, \emptyset, \emptyset, \emptyset)\) and \((\emptyset, 0, 20, 31, 40, \emptyset, \emptyset, m_{34}, \emptyset, \emptyset, \emptyset)\), but not in any \(\ell^*\)-reachable global state (see Figure 5.5). Also, under the assumption that all simplex channels in the protocol are one-slot buffers, one may check that two bo-pairs \((30, m_{34})\) and \((40, m_{43})\) exist which do not emerge in the \(\ell^*\)-reachability graph. In both cases the riddle relates to process \(P_3\) and process \(P_4\), which are always forced to progress in parallel as they are not indexed in any of the wait-sets of the \(\ell^*\)-reachable global states (i.e. they always have an executable transition and no potentially executable transitions in these states).

In order to identify the crux of the matter, let us look at two scenarios. Regarding unspecified receptions, suppose we want to find all ur-pairs for a process \(P_i\) and specifically those that involve a message over simplex channel \(C_{ji}\). Let \(G\) be the current global state and let \(y\) be the first message in \(C_{ji}\) if \(C_{ji}\) is not empty in \(G\). Clearly, we can determine whether \((s^{G}_i, y)\) is a ur-pair for \(P_i\) in \(G\) (i.e. a reception of \(y\) either is or is not defined at \(s^{G}_i\)) but, regardless, further progress of \(P_i\) is required to enable the detection of possible other ur-pairs for \(P_i\) with respect to \(C_{ji}\) in global states reachable from \(G\). Put differently, in this case process \(P_i\) need not be forced to wait in \(G\) unless it has potentially executable transitions at \(G\), as was the case before. The complication arises when channel \(C_{ji}\) is empty in \(G\). It may then be the case that process \(P_j\) can send a message \(y\) to \(P_i\), yielding a global state \(H\) with \(s^H_i = s^G_i\) and \(c^H_{ji} = y\), while \((s^G_i, y)\) is a ur-pair for \(P_i\) in \(H\) (i.e. no reception of \(y\) is defined at \(s^G_i\)). Yet, with the \(\ell^*\)-reachability relation \(P_i\) and \(P_j\) are forced to progress concurrently when \(i, j \not\in wait(G)\). That is, process \(P_i\) need not remain at \(s^G_i\) and hence the ur-pair \((s^G_i, x)\) may not be detected.

Regarding buffer overflows, suppose similarly that the objective is to find all bo-pairs which involve a message over a (prebounded) channel \(C_{ij}\). For a global state \(G\), let \(|c^G_{ij}| = B_{ij} - 1\) (i.e. \(C_{ij}\) can hold exactly one more message) and \(t_1 = (s^G_j, \neg x_1, s^H_i) \parallel X_{ij}(G)\) (i.e. \(G \parallel H\)). In addition, let \(t_2 = (s^H_i, \neg x_2, s) \parallel P_i(H)\) and \(t_3 = (s^G_j, +y, s) \parallel X_{ji}(G)\). Since \(t_2\) is potentially executable at \(H\), \((s^H_i, \neg x_2)\) is a bo-pair for \(P_i\) in \(H\). If transitions \(t_1\) and \(t_3\) are executed concurrently at \(G\), yielding a global state \(H\) then \(|c^G_{ij}| = B_{ij} - 1\) (\(C_{ij}\) still has one slot available) and hence \(t_2\) is executable at \(H\). In effect, we then leap over global state \(H\) and may thereby miss bo-pair \((s^H_i, \neg x_2)\). To detect it, process \(P_j\) must remain at \(s^G_j\) in \(G\) in particular because it has an executable receive transition pertaining to channel \(C_{ij}\).

We conclude from the above scenarios that, in general, special attention must be given to a process when it has an incoming empty channel or an executable receive transition at the current global state. More precisely, in order to detect the ur-pairs (with respect to a selected subset \(J\) of channels) in a protocol, the processes with an empty incoming channel (in \(J\)) in a global state \(G\) should be treated the same way as processes with potentially executable transitions at \(G\), i.e. they should be forced to wait in \(G\). Likewise, bo-pairs (with respect to a subset \(K\) of channels) can be
detected by forcing the processes with an executable receive transition at \( G \) (from a channel in \( K \)) to wait in \( G \).

**Definition 5.29**
Let \( J = \{ P_i \mid i \in I \}, L \) be a protocol and \( J, K \subseteq L \). A ur-pair \((s, y)\) for process \( P_i \) with \( y \in M_{ij} \) is said to be a ur-pair with respect to (wrt) \( J \) iff \( (j, i) \in J \). A bo-pair \((s, x)\) for process \( P_i \) with \( x \in M_{ij} \) is said to be a bo-pair wrt \( K \) iff \((i, j) \in K \).

Thus, when the aim is to identify all ur-pairs (bo-pairs) for a certain process \( P_i \), the index set \( J (K) \) should include every incoming (outgoing) channel of \( P_i \).

### 5.4.1 \( \ell(J, K)^* \)-reachability

In this section the \( \ell^* \)-reachability relation is parameterized with the channel-index sets \( J \) and \( K \). In analogy with definitions 5.4 and 5.16, we define the sets \( p\text{leap}(G, J, K) \) and \( x\text{pleap}(G, J, K) \) on the basis of a wait-set \( \text{wait}(G, J, K) \) which identifies not only the processes without executable transitions or with potentially executable transitions at \( G \) (as before), but also the processes with an incoming channel indexed in \( J \) that is empty in \( G \) and those with an executable receive transition at \( G \) from a channel indexed in \( K \).

**Definition 5.30**
Let \( G \) be a global state of a protocol \( J = \{ P_i \mid i \in I \}, L \) and \( J, K \subseteq L \). Define \( \text{wait}(G, J, K) = \{ i \in I \mid X_i(G) \neq \emptyset \} \) and

\[
\begin{align*}
\text{pleap}(G, J, K) &= \{ T \mid T \text{\leap}(G) \text{\act}(T) = \{ i \in I \mid i \text{\wait}(G, J, K) \} \} \\
&\quad \text{if } \text{wait}(G, J, K) \subseteq I \\
\text{pleap}(G, J, K) &= \{ \{ t \} \mid t \subseteq X(G) \} \\
&\quad \text{otherwise.}
\end{align*}
\]

**Definition 5.31**
Let \( G \) be a global state of a protocol \( J \) and \( T \) an arbitrary element of \( \text{pleap}(G, J, K) \). Define

\[
\begin{align*}
\text{xpleap}(G, J, K) &= \text{pleap}(G, J, K) \cup \{ T \cup \{ t \} \mid t \subseteq X(G) \text{\act}(t) \subseteq \text{wait}(G, J, K) \} \\
&\quad \text{if } \text{wait}(G, J, K) \subseteq I \\
\text{xpleap}(G, J, K) &= \text{pleap}(G, J, K) \\
&\quad \text{otherwise.}
\end{align*}
\]
Example 5.32
For the protocol of Example 5.6, let $G = (\emptyset 0, 21, 31, 41 \downarrow \uparrow m_{23}, m_{34}, \downarrow m_{43})$ be a (reachable) global state, then $X(G) = \{ t_1^1, t_2^3, t_4^2 \}$ and $\text{wait}(G, \{ (2, 3), (4, 3) \}, \{ (3, 4) \}) = \{ 1, 2, 4 \}$ since process $P_1$ has a potentially executable transition at $G$ (viz. $t_1^1$), $P_2$ has no executable transitions at $G$, and $P_4$ has an executable receive transition at $G$ involving channel $C_{34}$. Even though both incoming channels $C_{23}$ and $C_{43}$ of process $P_3$ are indexed, $P_3$ is not in the wait-set as both these channels are non-empty in $G$. Thus, we have $\text{pleap}(G, \{ (2, 3), (4, 3) \}, \{ (3, 4) \}) = \{ \{ t_2^3 \} \}$ and $\text{xpleap}(G, \{ (2, 3), (4, 3) \}, \{ (3, 4) \}) = \{ \{ t_2^3, t_1^1 \}, \{ t_2^3, t_4^2 \} \}$.

Again, without affecting the upcoming results we regard $\text{xpleap}(G, J, K)$ as a unique set by selecting for extension always the first element of $\text{pleap}(G, J, K)$ when viewed as an ordered set. Note that $\text{wait}(G, \emptyset, \emptyset) = \text{wait}(G)$ and hence $\text{pleap}(G, \emptyset, \emptyset) = \text{pleap}(G)$. Further note that $J \uparrow J\uparrow$ and $K \uparrow K\uparrow$ implies $\text{wait}(G, J, K) \uparrow \text{wait}(G, J\uparrow K\uparrow)$ but not $\text{pleap}(G, J, K) \uparrow \text{pleap}(G, J\uparrow K\uparrow)$ nor $|\text{pleap}(G, J, K)| < |\text{pleap}(G, J\uparrow K\uparrow)|$. Thus, in particular $\text{pleap}(G)$ and therefore $\text{xpleap}(G)$ are not necessarily subsets of respectively $\text{pleap}(G, J, K)$ and $\text{xpleap}(G, J, K)$. On the other hand, it is not hard to see that the properties of $\text{pleap}(G)$ and $\text{xpleap}(G)$ in propositions 5.5 and 5.17 hold similarly for $\text{pleap}(G, J, K)$ and $\text{xpleap}(G, J, K)$. Only those of $\text{xpleap}(G, J, K)$ are formulated for later reference.

Proposition 5.33
i) $t \uparrow X(G) \uparrow T \uparrow \text{xpleap}(G, J, K) \uparrow t \uparrow T$;

ii) $T \uparrow \text{xpleap}(G, J, K) \uparrow i \uparrow \text{act}(T) \uparrow \text{wait}(G, J, K) \uparrow t \uparrow X_i(G) \uparrow (T \setminus X_i(G)) \uparrow \{ t \} \uparrow \text{xpleap}(G, J, K)$;

iii) $T \uparrow \text{xpleap}(G, J, K) \uparrow i \uparrow \text{act}(T) \uparrow \text{wait}(G, J, K) \uparrow T \setminus X_i(G) \uparrow \text{pleap}(G, J, K)$  $\text{xpleap}(G, J, K) = \bigcup_{G \in \mathcal{X}(G)} \{ l \}$.  

Proof: Straightforward from definitions 5.30 and 5.31.

Naturally, the set $\text{xpleap}(G, J, K)$ induces a parameterized reachability relation, referred to as $\ell(J, K)^*$-reachability.

Definition 5.34
Let $G$ and $H$ be global states of a protocol $\mathcal{G}$. $G \uparrow (J, K) \uparrow H$ iff $T \uparrow \text{xpleap}(G, J, K) \uparrow \text{lin}(T)$ such that $G \uparrow \ell(J, K)^* H$. This is also denoted by $G \uparrow (J, K) \uparrow H$.

Definition 5.35
Let $G$ and $H$ be global states of a $\mathcal{G}$, and denote by $\uparrow (J, K)^*$ the reflexive and transitive closure of
Proof: Let \( \ell(J, K)^* \)-reachable from \( G \) \( \triangleright_{(J, K)}^{(\ell)} H \). If \( G = G^0 \), then \( H \) is said to be \( \ell(J, K)^* \)-reachable. The set of \( \ell(J, K)^* \)-reachable global states of \( \mathcal{J} \) is denoted by \( \mathbf{L}(J, K)_{\mathcal{J}}^\circ \). For a sequence of leap sets \( \mathcal{J} = T_1 T_2 \ldots T_m \), \( G \triangleright_{(J, K)}^{(\ell)} H \) denotes the existence of global states \( Q^0, Q^1, \ldots, Q^m \) such that \( G = Q^0 \triangleright_{(J, K)}^{(\ell)} Q^1 \triangleright_{(J, K)}^{(\ell)} \ldots \triangleright_{(J, K)}^{(\ell)} Q^m = H \).

**Proposition 5.36**

\[ \mathbf{L}(J, K)_{\mathcal{J}}^\circ \succ \mathbf{R}_{\mathcal{J}} \]

**Proof:** By definition of \( \triangleright_{(J, K)}^{(\ell)} \).

It is clear that \( \mathbf{L}(\emptyset, \emptyset)_{\mathcal{J}}^\circ = \mathbf{L}_{\mathcal{J}}^\circ \) and thus \( \mathbf{L}(\emptyset, \emptyset)_{\mathcal{J}}^\circ \) reveals all non-progress states and all non-executable transitions of a protocol (by Corollary 5.22 and Theorem 5.26). As we will show, the same holds true for \( \mathbf{L}(J, K)_{\mathcal{J}}^\circ \) in general, but note that this cannot be established directly since \( \mathbf{L}_{\mathcal{J}}^\circ \) need not be included in \( \mathbf{L}(J, K)_{\mathcal{J}}^\circ \) (i.e. \( \text{xpleap}(G) \) is generally not a subset of \( \text{xpleap}(G, J, K) \)).

### 5.4.2 Detecting ur-pairs and bo-pairs

We continue by proving that state exploration based on \( \triangleright_{(J, K)}^{(\ell)} \) preserves all non-progress states, all (non-)executable transitions, all unspecified receptions wrt \( J \) and all buffer overflows wrt \( K \).

**Lemma 5.37**

Let \( G \triangleright_{(J, K)}^{(\ell)} H \) and \( \mathcal{J} \neq \mathcal{J} \) then there exist a leap set \( T \triangleright_{(J, K)}^{(\ell)} \) with \( \mathcal{J} \triangleright_{(J, K)}^{(\ell)} \text{lin}(T) \), transition sequences \( \mathcal{J}, \mathcal{J} \) and a global state \( H \mathcal{J} \) such that

i) \( \mathcal{J} \triangleright_{(J, K)}^{(\ell)} \), \( \text{act}(\mathcal{J}) \triangleright_{(J, K)}^{(\ell)} \text{act}(\mathcal{J}) = \emptyset \) and \( |\mathcal{J}| < |\mathcal{J}| \);

ii) \((s, y)\) is a ur-pair wrt \( J \) in \( H \triangleright_{(J, K)}^{(\ell)} \) \((s, y)\) is a ur-pair (wrt \( J \)) in \( G \) or in \( H \triangleright_{(J, K)}^{(\ell)} \);

iii) \((s, x)\) is a bo-pair wrt \( K \) in \( H \triangleright_{(J, K)}^{(\ell)} \) \((s, x)\) is a bo-pair (wrt \( K \)) in \( G \) or in \( H \triangleright_{(J, K)}^{(\ell)} \).

**Proof:**

i) Akin to the proof of Lemma 5.24 (and Lemma 5.10), mutatis mutandis. That is, substitute \( \text{xpleap}(G, J, K) \) for \( \text{xpleap}(G) \) and \( \text{xpleap}(G, J, K) \) for \( \text{xpleap}(G) \). Observe from this proof that \( T \triangleright_{(J, K)}^{(\ell)} \) \( \text{lin}(T) \) remains such that \( T \setminus \text{pref}(\mathcal{J}) \) is minimal and \( \mathcal{J} \triangleright_{(J, K)}^{(\ell)} \) \( \text{lin}(T \setminus \text{pref}(\mathcal{J})) \), with \( \text{first}(\mathcal{J}) \) = \( \{ t \mid t \in T, \mathcal{J} \triangleright_{(J, K)}^{(\ell)} \text{pref}(\mathcal{J}) \triangleright_{(J, K)}^{(\ell)} \text{act}(\mathcal{J}) \} \).

ii) By definition, if \((s, y)\) is a ur-pair wrt \( J \) in \( H \) then \( s = s_i^H, y \triangleright_{M_{ji}} \) and \( \text{front}(c_i^H) = y \), for some \((j, i) \triangleright_{(J, K)}^{(\ell)} \). If \( i \triangleright_{(J, K)}^{(\ell)} \text{act}(\mathcal{J}) \), then since \( H \triangleright_{(J, K)}^{(\ell)} \), we have \( s_i^{\mathcal{J}} = s_i^H \) and thus \((s_i^H, y)\) is a ur-pair in \( H \triangleright_{(J, K)}^{(\ell)} \). Alternatively, if \( i \triangleright_{(J, K)}^{(\ell)} \text{act}(\mathcal{J}) \) then \( i \triangleright_{(J, K)}^{(\ell)} \text{act}(\mathcal{J}) \) and thus \( s_i^H = s_i^\mathcal{J} \). Three cases must then be considered:
• \( c_{ji}^G = \square \)
Since \( i \not\in \text{act}(\square) \) and \((j, i) \not\in J \) we have \( i \not\in \text{act}(T \not\square) \) \text{wait}(G, J, K) \) and consequently, by Proposition 5.33.(iii), \( T \setminus X_j(G) \not\square \text{pleap}(G, J, K) \) or \( \text{xpleap}(G, J, K) = \bigcup_{i \in X_j(G)} \{t\} \). Either case contradicts the minimality of \( T \setminus \text{first}(\square) \);
• \( \text{front}(c_{ji}^G) = y \)
Since \((s_i^y, y) \) is a ur-pair in \( H \) and \( s_i^y = s_i^G \), \((s_i^y, y) \) is also a ur-pair in \( G \);
• \( \text{front}(c_{ji}^G) = z, \) with \( z \not= y \)
Since \( i \not\in \text{act}(\square) \) we have \( \text{front}(c_{ji}^G) = z \), which contradicts the fact that \( \text{front}(c_{ji}^G) = y \).

In conclusion, \((s, y) \) is a ur-pair wrt \( J \) in \( G \) or in \( H \)

iii) By definition, if \((s, x) \) is a bo-pair wrt \( K \) in \( H \) then \( s = s_i^H, x \not\square M_{ij} \) and \( \left| c_{ij}^H \right| = B_{ij}, \) for some \((i, j) \not\in K \). Three cases are considered:
• \( i \not\in \text{act}(\square) \)
In this case, \( s_i^H = s_i^G \) and \( \left| c_{ij}^H \right| = B_{ij} \) (no message is sent to \( C_{ij} \) or received from \( C_{ij} \) along \( \square \), since otherwise \( i \not\in \text{act}(\square) \) or \( \left| c_{ij}^H \right| < B_{ij} \)). Thus, \((s_i^H, x) \) is a bo-pair in \( G \);
• \( i \not\in \text{act}(\square), j \not\in \text{act}(\square) \)
Since \( \text{act}(\square) \not\square \text{act}(\square) = \emptyset \) we have \( i \not\in \text{act}(\square) \). Clearly, \( H \not\square \text{xpleap}(G) \) and \( i, j \not\in \text{act}(\square) \) imply that \( s_i^H = s_i^H \) and \( \left| c_{ij}^H \right| = B_{ij} \), i.e. \((s, x) \) is a bo-pair in \( H \)
• \( i \not\in \text{act}(\square), j \not\in \text{act}(\square) \)
Again, \( i \not\in \text{act}(\square) \) implies \( i \not\in \text{act}(\square) \). Since \( \not\square \text{lin}(T \setminus \text{first}(\square)) \) and \( T \setminus \text{first}(\square) \not\square \text{leap}(G) \), there is exactly one transition \( t \) from process \( P_j \) in \( \square \) and \( t \not\square X_j(G) \). Clearly, if \( t \) does not entail a reception from channel \( C_{ij} \), then \((s_i^H, x) \) is a bo-pair in \( H \). On the other hand, if \( t \) does entail a reception from \( C_{ij} \) we have \( j \not\in \text{act}(T) \not\square \text{wait}(G, J, K) \) since \( j \not\in \text{act}(\square) \) and \((i, j) \not\in K \). Again, \( T \setminus X_j(G) = (T \setminus \{t\}) \not\square \text{pleap}(G, J, K) \) or \( \text{xpleap}(G, J, K) = \bigcup_{i \in X_j(G)} \{t\} \), by Proposition 5.33.(iii), and either case contradicts the minimality of \( T \setminus \text{first}(\square) \).

In conclusion, \((s, x) \) is a bo-pair wrt \( K \) in \( G \) or in \( H \)

\[ \square \]

\textbf{Lemma 5.38}\]

Let \( G \not\square \text{xpleap}(G) \) and \( \square \not= \square \) then there exist a sequence of leap sets \( \square \) with \( \square \not\square \text{lin}(\square) \), a transition sequence \( \square \) and a global state \( H \) such that \( G \not\square \text{lin}(H, G) \not\square \text{xpleap}(G) \) and

i) \( \square \not\square \text{lin}(\square) \);
ii) \((s, y) \) is a ur-pair wrt \( J \) in \( H \not\square \square \text{pref}(\square) \): \( G \not\square \text{xpleap}(G) \not\square \text{xpleap}(G) \) and \((s, y) \) is a ur-pair in \( G \)
iii) \((s, x) \) is a bo-pair wrt \( K \) in \( H \not\square \square \text{pref}(\square) \): \( G \not\square \text{xpleap}(G) \not\square \text{xpleap}(G) \) and \((s, x) \) is a bo-pair in \( G \)

\textbf{Proof:} Repeated application of Lemma 5.37.(i) yields the following diagram (cf. Lemma 5.25):
Let \( \square = T_1 T_2 \ldots T_m \) with \( \square \subseteq \text{lin}(\square) \), \( \square = \square_1 \square_2 \ldots \square_m \) and \( H^0 = H^m \), then \( G \equiv_{(J,K)^*} H^0 \text{ and } \square \equiv_{(J,K)^*} G^m = H^m \).

Assume that \((s, y)\) is a ur-pair wrt \( J \) in \( H \), then by Lemma 5.37.(ii), \((s, y)\) is a ur-pair wrt \( J \) in \( G \) or in \( H^1 \). The claim trivially holds if \((s, y)\) is a ur-pair wrt \( J \) in \( G \) (let \( \square \equiv = \square \)). On the other hand, if \((s, y)\) is a ur-pair wrt \( J \) in \( H^1 \), then again by Lemma 5.37.(ii) \((|\square_1| > 0)\), \((s, y)\) must be a ur-pair wrt \( J \) in \( G^1 \) or in \( H^2 \). By repeating this argument it follows easily that \((s, y)\) must be a ur-pair wrt \( J \) in (at least) one of the \( (J, K)^*\)-reachable global states \( G^1, G^2, \ldots, G^m \), say in \( G^j \), in particular because \( G^m = H^m \). The claim then holds by choosing \( \square \equiv = T_1 T_2 \ldots T_j \).

iii) Analogous to the proof of part (ii), using Lemma 5.37.(iii). \( \square \)

Note the resemblance between Lemma 5.38.(i) and Lemma 5.25. As a consequence, the results established for \( L^*_J \) also hold for \( L(J,K)_J^* \), viz. exploring the \( (J, K)^*\)-reachability graph of a protocol suffices to detect all non-progress states and all non-executable transitions (independent of \( J \) and \( K \)). Lemma 5.38.(ii) and (iii) yield the result anticipated for unspecified receptions and buffer overflows.

**Theorem 5.39**

Every non-progress state is \( (J, K)^*\)-reachable, for all \( J, K \equiv L \).

**Proof:** Let \( H \) be a non-progress state with \( G^0 \equiv_{(J,K)^*} H \). \( H \) is trivially \( (J, K)^*\)-reachable if \( |\square| = 0 \). If \( |\square| > 0 \), then by Lemma 5.38.(i) and the fact that \( H \) is a non-progress state (i.e. \( \square = \square \)), there is a sequence of leap sets \( \square \) with \( \square \subseteq \text{lin}(\square) \) such that \( G^0 \equiv_{(J,K)^*} H \) and \( \square \equiv_{(J,K)^*} H \). Again, \( H \) is \( (J, K)^*\)-reachable. \( \square \)

**Theorem 5.40**

A transition \( t \) is executable iff \( t \) is executable at an \( (J, K)^*\)-reachable global state, for all \( J, K \equiv L \).

**Proof:** The “if” part holds directly since \( L(J,K)_J^* \equiv R_J \). For the “only-if” part, when \( t \) is executable there exists a global state \( H \) such that \( G^0 \equiv (J,K)^* H \), for some transition sequence \( \square \). By Lemma 5.38.(i), there exist \( \square \) with \( \square \subseteq \text{lin}(\square) \), \( \square \) and \( H \) such that \( G^0 \equiv_{(J,K)^*} H \) and \( \square \equiv_{(J,K)^*} H \). Hence, since \( t \) appears in \( \square \) it must be executable at an \( (J, K)^*\)-reachable global state. \( \square \)
Theorem 5.41
If \( (s, y) \) is a ur-pair wrt \( J \), then \( (s, y) \) is a ur-pair in an \( \ell(J, K)^* \)-reachable global state, for all \( K \subseteq L \).

Proof: By definition, \( (s, y) \) is a ur-pair wrt \( J \) in some reachable global state \( H \). Let \( G^0 \xrightarrow{\ell(J, K)^*} H \) for some transition sequence \( \emptyset \). The theorem holds trivially if \( |\emptyset| = 0 \). For the case \( |\emptyset| > 0 \), the proof is immediate from Lemma 5.38.(ii).

Theorem 5.42
If \( (s, x) \) is a bo-pair wrt \( K \), then \( (s, x) \) is a bo-pair in an \( \ell(J, K)^* \)-reachable global state, for all \( J \subseteq L \).

Proof: By definition, \( (s, x) \) is a bo-pair wrt \( K \) in some reachable global state \( H \). Let \( G^0 \xrightarrow{\ell(J, K)^*} H \) for some transition sequence \( \emptyset \). The theorem holds trivially if \( |\emptyset| = 0 \). For the case \( |\emptyset| > 0 \), the proof is immediate from Lemma 5.38.(iii).

The detection of all unspecified receptions in a protocol is thus guaranteed by choosing \( J = L \) and \( K \) arbitrary, and vice versa for buffer overflows. Of course, for protocols whose channels are not prebounded the detection of buffer overflows is immaterial and \( K \) should be set to empty.

Corollary 5.43
For a protocol \( \emptyset \), indefinite progress, freedom of non-executable transitions, and freedom of unspecified receptions (wrt \( J \)) and buffer overflows (wrt \( K \)) are decidable if \( L(J, K)^*_{\emptyset} \) is finite.

It is obvious from the presentation that the task of detecting unspecified receptions and buffer overflows in a protocol can be performed via multiple independent subtasks, simply by partitioning the index set \( L \) of all channels. Surely, this is an important feature of the parameterized reachability relation \( \xrightarrow{\ell(J, K)}^* \) since in many cases each subtask utilizes a smaller number of \( \ell(J, K)^* \)-reachable global states and transitions than the corresponding full task. The memory needed for verification may then be reduced further by executing the subtasks in sequence on a single processor, whereas the time consumption may decrease by executing them in parallel at the expense of using multiple processors.

Example 5.44
Figure 5.6 shows in part the \( \ell(J, K)^* \)-reachability graph of the protocol of Example 5.6 for \( J = L = \{ (1, 2), (2, 3), (3, 4), (4, 1), (4, 3) \} \) and \( K = \emptyset \) (empty channels in the states are omitted and dashed arrows indicate incomplete paths). For each \( \ell(L, \emptyset)^* \)-reachable global state \( G^k \) in the figure, Table 5.2 lists the data used to calculate the set \( xpleap(G^k, L, \emptyset) \) of leap sets executed at \( G^k \). The elements of \( xpleap(G^k, L, \emptyset) \) are precisely the labels of the outgoing edges of the node labeled \( G^k \). The complete \( \ell(L, \emptyset)^* \)-reachability graph of the protocol of Example 5.6 consists of 29 nodes and 69
edges versus respectively 40 and 100 for the conventional reachability graph. State exploration by LRA based on \( \mathcal{O} \) correctly determines that the protocol has no non-progress states and one non-executable transition, viz. \( t_1^2 = (10, +m_{41}, 12) \). It also reveals all unspecified receptions, viz. \((21, m_{12}), (30, m_{23}), (30, m_{43}), (31, m_{23}) \) and \((40, m_{34})\).

To illustrate the extra space reduction that can be obtained by dividing the task of detecting all unspecified receptions into several independent subtasks, consider the partitioning of \( J = L \) into the disjoint subsets \( J_1 = \{(4, 1), (1, 2)\} \) (i.e. the incoming channels of processes \( P_1 \) and \( P_2 \)), \( J_2 = \)

![Diagram](image)

**Figure 5.6** The partial \( r(L, \emptyset)^s \)-reachability graph of the protocol of Example 5.6.

**Table 5.2** Data for Figure 5.6.

<table>
<thead>
<tr>
<th>Global state ( G^k )</th>
<th>( X(G^k) )</th>
<th>( P(G^k) )</th>
<th>( \text{wait}(G^k, L, \emptyset) )</th>
<th>( \text{pleap}(G^k, L, \emptyset) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G^0 )</td>
<td>{ ( t_1^1, t_2^2, t_3^3, t_4^4 ) }</td>
<td>{ ( t_2^1, t_2^2 ) }</td>
<td>( I )</td>
<td>{ ( { t_1^1 }, { t_2^2 }, { t_3^3 }, { t_4^4 } ) }</td>
</tr>
<tr>
<td>( G^1 )</td>
<td>{ ( t_1^1, t_2^2, t_4^4 ) }</td>
<td>{ ( t_2^1, t_2^2, t_2^3 ) }</td>
<td>{1, 2, 3}</td>
<td>{ ( { t_2^2 } ) }</td>
</tr>
<tr>
<td>( G^2 )</td>
<td>{ ( t_1^1, t_2^2, t_1^1 ) }</td>
<td>{ ( t_2^1, t_2^2, t_2^3 ) }</td>
<td>( I )</td>
<td>{ ( { t_1^1 }, { t_2^2 }, { t_3^3 } ) }</td>
</tr>
<tr>
<td>( G^3 )</td>
<td>{ ( t_1^1, t_2^2, t_3^3, t_4^4 ) }</td>
<td>( \emptyset )</td>
<td>{1, 3}</td>
<td>{ ( { t_1^1, t_2^2 }, { t_3^3, t_4^4 } ) }</td>
</tr>
<tr>
<td>( G^4 )</td>
<td>{ ( t_1^1, t_4^4 ) }</td>
<td>{ ( t_2^1, t_2^3 ) }</td>
<td>{1, 2, 4}</td>
<td>{ ( { t_2^3 } ) }</td>
</tr>
<tr>
<td>( G^5 )</td>
<td>{ ( t_3^3, t_4^4 ) }</td>
<td>( \emptyset )</td>
<td>{1, 2, 4}</td>
<td>{ ( { t_3^3 } ) }</td>
</tr>
<tr>
<td>( G^6 )</td>
<td>{ ( t_1^1, t_2^2, t_4^4 ) }</td>
<td>{ ( t_2^1 ) }</td>
<td>{1, 2}</td>
<td>{ ( { t_2^1 }, { t_4^4 } ) }</td>
</tr>
</tbody>
</table>
{(2, 3), (4, 3)} (i.e. the incoming channels of process $P_3$) and $J_3 = \{(3, 4)\}$ (i.e. the incoming channel of process $P_4$). We obtain the following results:

- the $\ell(J_1, \varnothing)^*$-reachability graph consists of 10 nodes and 18 edges (this graph is in fact the same as the $\ell^*$-reachability graph in Figure 5.5) – three ur-pairs are detected, including all those for processes $P_1$ and $P_2$ (if any): $(21, m_{12})$, $(30, m_{23})$, $(31, m_{23})$;
- the $\ell(J_2, \varnothing)^*$-reachability graph consists of 22 nodes and 51 edges – four ur-pairs are detected, including all those for process $P_3$: $(21, m_{12})$, $(30, m_{23})$, $(31, m_{23})$ and $(30, m_{43})$;
- the $\ell(J_3, \varnothing)^*$-reachability graph consists of 15 nodes and 32 edges – four ur-pairs are detected, including all those for process $P_4$: $(21, m_{12})$, $(30, m_{23})$, $(31, m_{23})$ and $(40, m_{34})$.

All five unspecified receptions are thus detected and since the $\ell(J_2, \varnothing)^*$-reachability graph is the largest one constructed, it represents the total space required for this analysis. Only 22 instead of 29 nodes need be stored, i.e. additional space reduction is achieved at the expense of additional running time (when using only one processor).

**Example 5.45**

Consider the protocol of Example 5.6 with a capacity bound of one message for each channel. Its $\ell(J, K)^*$-reachability graph is shown in part in Figure 5.7 for $J = \varnothing$ and $K = L$ (empty channels in the states are omitted and dashed arrows indicate incomplete paths). For every $\ell(\varnothing, L)^*$-reachable global state $G^k$ in the figure, Table 5.3 provides the data used to calculate $xpleap(G^k, \varnothing, L)$, the elements of which match the labels of the outgoing edges of the node labeled $G^k$. The complete

![Diagram](image-url)

**Figure 5.7** The partial $\ell(\varnothing, L)^*$-reachability graph of the protocol of Example 5.6 (with $B_{ij} = 1$ for all $(i, j) \in L$).
reachability graph. This cannot be done in terms of the set \( X(G) \).

In this section we explicitly assume a depth-first search, but not yet expanded during the search, as in depth-first order as well as in breadth-first order (given \( m \)).

In particular, the leaping search has been turned into an opportunity to refine the \( \ell(J, K)^* \)-reachability relation. Essentially, it warrants a more accurate solution to the ignoring problem than the one given earlier in terms of the set \( xpleap(G, J, K) \).

Recall from the beginning of Section 5.3 that the ignoring problem occurs when the progress of some processes is indefinitely deferred along a cycle in the \( \ell \)-reachability graph. This cannot happen in the \( \ell(J, K)^* \)-reachability graph since it is guaranteed that \( \bigcup_{T \subseteq xpleap(G, J, K)} T = X(G) \) for every \( \ell(J, K)^* \)-reachable global state \( G \) (cf. Proposition 5.33(i)).

### Table 5.3 Data for Figure 5.7.

<table>
<thead>
<tr>
<th>Global state ( G )</th>
<th>( X(G) )</th>
<th>( P(G) )</th>
<th>( \text{wait}(G, \emptyset, L) )</th>
<th>( \text{pleap}(G, \emptyset, L) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G^0 )</td>
<td>( {t_1^1, t_1^2, t_1^3, t_1^4} )</td>
<td>( {t_1^2, t_1^3} )</td>
<td>( {1, 2} )</td>
<td>( {(t_1^1, t_1^4)} )</td>
</tr>
<tr>
<td>( G^1 )</td>
<td>( {t_2^1, t_2^2, t_2^3, t_2^4} )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( {(t_2^1, t_2^3), (t_2^2, t_2^4)} )</td>
</tr>
<tr>
<td>( G^2 )</td>
<td>( {t_1^1, t_1^2, t_1^3, t_1^4} )</td>
<td>( {t_1^2, t_1^3} )</td>
<td>( \emptyset )</td>
<td>( {(t_1^1, t_1^3), (t_2^2, t_2^4)} )</td>
</tr>
<tr>
<td>( G^3 )</td>
<td>( {t_1^1, t_1^2, t_1^3, t_1^4} )</td>
<td>( {t_1^2} )</td>
<td>( \emptyset )</td>
<td>( {(t_1^1), (t_2^2) } )</td>
</tr>
<tr>
<td>( G^4 )</td>
<td>( {t_1^1, t_1^2, t_1^3, t_1^4} )</td>
<td>( {t_1^2, t_1^3} )</td>
<td>( \emptyset )</td>
<td>( {(t_1^1), (t_2^3) } )</td>
</tr>
<tr>
<td>( G^5 )</td>
<td>( {t_1^1, t_1^2, t_1^3} )</td>
<td>( {t_1^2, t_1^3} )</td>
<td>( \emptyset )</td>
<td>( {(t_1^1), (t_2^3) } )</td>
</tr>
</tbody>
</table>

\( \ell(\emptyset, L)^* \)-reachability graph of the protocol of Example 5.6 consists of 20 nodes and 45 edges versus respectively 30 and 70 for the conventional reachability graph (which is different from the previous examples due to the imposed channel bounds). Additional reductions are obtained when \( K = L \) is partitioned into disjoint subsets, similar as in Example 5.44. Once again, the protocol is found to have no non-progress states and one non-executable transition. Only three of five ur-pairs are detected in this case, which is not unexpected since \( J = \emptyset \). Lastly, two bo-pairs \((30, m_{34})\) and \((40, m_{43})\) are detected as well, indicating that channels \( C_{34} \) and \( C_{43} \) require a bound larger than one (a bound of two messages turns out to be sufficient here).

\( \square \)

### 5.5 More reduction with a depth-first search

Thus far in this chapter no specific search technique has been assumed in the formulation of LRA. In particular, the \( \ell(J, K)^* \)-reachable (or \( \ell \)-reachable) global state space of a protocol can be searched in depth-first order as well as in breadth-first order (given of course that the search space is finite).

As discussed in Chapter 2, a depth-first search uses a stack to trace the global states encountered but not yet expanded during the search, while a breadth-first search uses a queue for this purpose. In this section we explicitly assume a depth-first search (DFS) to accommodate a further reduction of the number of global states and transitions explored for detecting non-executable transitions, unspecified receptions and buffer overflows.

The use of a DFS can be turned into an opportunity to refine the \( \ell(J, K)^* \)-reachability relation. Essentially, it warrants a more accurate solution to the ignoring problem than the one given earlier in terms of the set \( xpleap(G, J, K) \).
thus in particular for all such states that constitute a cycle in the \( \ell(J, K)^* \)-reachability graph. Simple graph-based reasoning shows, however, that it is sufficient to have in each cycle just one global state with this property. Avoiding the ignoring problem then becomes a matter of detecting cycles during state exploration, for which a DFS lends itself preeminently: a cycle is detected precisely when a global state reached from the current global state already resides on the DFS stack [CLR90, Pel96]. This is translated into an extra condition for extending sets of proper leap sets, viz. for a global state \( G \) the set \( \text{pleap}(G, J, K) \) is extended only if (1) \( \text{wait}(G, J, K) \subseteq I \) (as in Definition 5.31) and (2) at least one element of \( \text{pleap}(G, J, K) \) leads from \( G \) to a global state on the DFS stack.

**Definition 5.46**

Let \( G \) be a global state of a protocol \( \emptyset \) to be expanded during the DFS, and let \( T \) be an arbitrary element of \( \text{pleap}(G, J, K) \). Define

\[
x\text{pleap}-2(G, J, K) = \text{pleap}(G, J, K) \cup \{ T \mid \{ t \mid t \in \text{X}(G) \cup \text{act}(t) \cup \text{wait}(G, J, K) \} \}
\]

if \( \text{wait}(G, J, K) \subseteq I \) and \( [T] \in \text{pleap}(G, J, K) \): \( G \) has a \( \text{DFS} \)-reachable global state \( H \) such that \( H \) is on the DFS stack

\[
x\text{pleap}-2(G, J, K) = \text{pleap}(G, J, K)
\]

otherwise.

The reachability relation resulting from the execution of the leap sets in \( x\text{pleap}-2(G, J, K) \) in global states is referred to accordingly as \( \ell_2(J, K)^* \)-reachability and denoted by \( \l_2(J, K)^* \). The set of \( \ell_2(J, K)^* \)-reachable global states of a protocol \( \emptyset \) (i.e. \( \ell_2(J, K)^* \)-reachable from its initial global state) is denoted by \( \text{L}_2(J, K)^* \). When \( J = K = \emptyset \), we may drop these index sets from the notations, as in Section 5.4. Under the assumption that it takes constant time to check whether a global state is on the DFS stack (which can indeed be implemented efficiently by a simple hash-table look-up), the extra cost incurred for computing \( x\text{pleap}-2(G, J, K) \) instead of \( x\text{pleap}(G, J, K) \) is \( O(|\text{pleap}(G, J, K)|) \) in the worst case. On the other hand, it should be clear from the definitions that \( x\text{pleap}-2(G, J, K) \subseteq x\text{pleap}(G, J, K) \), for any global state \( G \). The \( \ell_2(J, K)^* \)-reachability graph of a given protocol is thus a subgraph of its \( \ell(J, K)^* \)-reachability graph. Furthermore, by construction, for every cycle in the \( \ell_2(J, K)^* \)-reachability graph there exists at least one global state \( G \) in that cycle for which \( \bigcup_{T \in x\text{pleap}(G, J, K)} T = \text{X}(G) \).

**Proposition 5.47**

\( \text{L}_2(J, K)^* \subseteq \text{L}(J, K)^* \)

**Proof:** Since \( x\text{pleap}-2(G, J, K) \subseteq x\text{pleap}(G, J, K) \) for any global state \( G \), it is immediate that every \( \ell_2(J, K)^* \)-reachable global state is \( \ell(J, K)^* \)-reachable.
The next two lemmas come in place of Lemma 5.37 and Lemma 5.38 in Section 5.4.2, proving that the \(\ell_2(J, K)^*\)-reachability graph of a protocol equally reveals all non-progress states, all non-executable transitions, all unspecified receptions wrt \(J\) and all buffer overflows wrt \(K\).

**Lemma 5.48**

Let \(G\) be a global state that is removed from the DFS stack during the construction of the \(\ell_2(J, K)^*\)-reachability graph, with \(G \not\in_{\ell_2(J, K)} \emptyset\) and \(\emptyset \neq \emptyset\) then there exist a sequence \(G \not\in_{\ell_2(J, K)} G_1 \not\in_{\ell_2(J, K)} \ldots \not\in_{\ell_2(J, K)} G_m\) with \(\emptyset \subseteq \text{lin}(T_i)\), transition sequences \(\emptyset\), \(\emptyset\) and a global state \(H\) such that

i) \(\emptyset \subseteq G \subseteq H_1 \subseteq \ldots \subseteq H_n \subseteq \emptyset\), \(\text{act}(\emptyset) \subseteq \emptyset\) and \(\emptyset \neq \emptyset\);

ii) \((s, y)\) is a ur-pair wrt \(J\) in \(H\), \((s, y)\) is a ur-pair wrt \(J\) in \(G\) or in \(H\);

iii) \((s, x)\) is a bo-pair wrt \(K\) in \(H\) (or \(s, x\) is a bo-pair (wrt \(K\) in \(G\) or in \(H\).

**Proof:** We prove only part (i). Part (ii) and (iii) are derived from part (i) in exactly the same way as Lemma 5.37.(ii) and (iii) are derived from Lemma 5.37.(i), particularly because the property of \(\text{xpleap}(G, J, K)\) stated by Proposition 5.33.(iii) also holds for \(\text{xpleap-2}(G, J, K)\) (cf. the proof of Lemma 5.37). Part (i) is visualized as follows:

\[
\begin{array}{ccc}
G & \xrightarrow{T_1, T_2, \ldots, T_n} & \ell_2(J, K)^* \emptyset \\
\downarrow & & \downarrow H^* \\
\emptyset & & \emptyset \\
\end{array}
\]

Analogous to the proof of Lemma 5.10, it can be shown that there exist \(T \subseteq \text{xpleap}(G, J, K) \subseteq \text{xpleap-2}(G, J, K)\) with \(\emptyset \subseteq \text{lin}(T)\), transition sequences \(\emptyset\) and \(\emptyset\) and a global state \(H\) such that \(\emptyset \subseteq G \subseteq H_1 \subseteq \ldots \subseteq H_n \subseteq \emptyset\), \(\text{act}(\emptyset) \subseteq \emptyset\) and \(\emptyset \neq \emptyset\). In particular, \(T\) and \(H\) are such that \(T\) \(\not\in \text{first}(\emptyset)\) is minimal and \(\emptyset \subseteq \text{lin}(T) \\text{pref}(\emptyset) \subseteq \emptyset \text{act}(\emptyset)\). Clearly, if \(T \not\in \text{first}(\emptyset)\), \(T \not\in \emptyset\) then \(\emptyset \neq \emptyset\) and thus \(\emptyset \neq \emptyset\). Consequently, in this case the lemma holds by letting \(m = 1, T_1 = T\), \(H = \emptyset\) and \(\emptyset = \emptyset\).

Alternatively, if \(T \not\in \text{first}(\emptyset)\), \(\emptyset \subseteq \text{lin}(T) \\text{pref}(\emptyset) \subseteq \emptyset \text{act}(\emptyset)\) and \(T \not\in \emptyset\). We must have \(\text{wait}(G, J, K) \subseteq \emptyset\) because otherwise \(\text{xpleap}(G, J, K) = \bigcup_{I \subseteq G_i} \{I\}\) and thus \(\emptyset \not\in \text{xpleap}(G, J, K)\), with the first transition of \(\emptyset\), but \(|\{I\} \\setminus \text{first}(\emptyset)| = 0 < |T \setminus \text{first}(\emptyset)|\) contradicting the minimality of \(T \not\in \text{first}(\emptyset)\). Definition 5.30 then states that \(\emptyset \subseteq \text{xpleap}(G, J, K)\): \(\text{act}(\emptyset) = \emptyset\) and since \(\emptyset \subseteq \text{lin}(T)\) it follows further that \(\emptyset \subseteq \text{xpleap}(G, J, K)\): \(\text{act}(\emptyset) \subseteq \emptyset\) and \(\text{act}(\emptyset) \subseteq \text{wait}(G, J, K)\). In particular the first transition \(\emptyset\) of \(\emptyset\) is therefore not in any element of \(\text{xpleap}(G, J, K)\). The proof now continues by induction on the order in which \(\ell_2(J, K)^*\)-reachable global states are removed from the DFS stack. Two cases can arise when removing \(G\) from the stack:
• some \( T \uparrow\text{pleap}(G, J, K) \) leads to a global state that is already on the DFS stack

Remark that this case covers in particular the induction basis where \( G \) is the first state removed from the DFS stack: each set in \( \text{pleap}(G, J, K) \) executed in \( G \) leads back to a global state on the DFS stack since any other global state would have been removed before \( G \) (a characteristic of a depth-first search). According to Definition 5.46, some element of \( \text{pleap}(G, J, K) \), say \( T'' \), is extended to form \( \text{xpleap-2}(G, J, K) \). Now choose \( T'_1 = T'' \uparrow \{ t \} \) (\( t \) is the first transition of \( \emptyset \)), then \( T'_1 \uparrow\text{xpleap-2}(G, J, K), T'_1 \setminus \text{first}(\emptyset) = T'' \setminus \text{first}(\emptyset) = T'' \) and \( \text{act}(T'_1) \uparrow \text{act}(\emptyset) = \text{act}(\emptyset) \). The lemma holds again by letting \( m = 1, (t_1 \uparrow \text{lin}(T'_1), \emptyset \uparrow \text{lin}(T'')) \) and \( \emptyset = t \emptyset \) viz. \( \emptyset \equiv_{H_0} (\emptyset \uparrow \{ t \} \) for some global state \( H \uparrow \text{act}(\emptyset) = \emptyset \) and \( \emptyset < \emptyset \).

• no \( T \downarrow \text{pleap}(G, J, K) \) leads to a global state that is already on the DFS stack

Let \( T_1 = T_1 \uparrow \text{and } G \equiv_{\{ r \mid J, K \}} G^1 \). Since \( T_1 \downarrow \text{pleap}(G, J, K) \), \( G^1 \) is not on the DFS stack and when added it will be removed before \( G \) itself is removed (a characteristic of a depth-first search). We also know that \( \text{act}(\emptyset) \downarrow \text{act}(T_1) = \emptyset \) and hence \( \emptyset \) can still be executed from \( G^1 \). But this means that the induction hypothesis can be applied to \( G^1 \), viz. there exist a sequence \( G^1 \equiv_{\{ r \mid J, K \}} \cdots \equiv_{\{ r \mid J, K \}} G^m \) with \( \emptyset \downarrow \text{lin}(T_1) \), transition sequences \( \emptyset \) and \( \emptyset \) and a global state \( H \uparrow \) such that \( \emptyset \equiv_{\{ r \mid J, K \}} \cdots \equiv_{\{ r \mid J, K \}} \text{act}(\emptyset) \uparrow \text{act}(\emptyset) = \emptyset \) and \( \emptyset < \emptyset \). It follows that \( \emptyset \equiv_{\{ r \mid J, K \}} \cdots \equiv_{\{ r \mid J, K \}} \text{act}(\emptyset) \uparrow \text{act}(\emptyset) = \emptyset \). The lemma thus holds with \( \emptyset = \emptyset \) and \( \emptyset = \emptyset \). 

Lemma 5.48 differs from Lemma 5.37 mainly in stipulating the existence of a sequence of leap sets that satisfies the three stated properties, instead of a single leap set. Notice indeed that the sequence sought in Lemma 5.48 is guaranteed to be of length one when applying the lemma to the \( \ell(J, K)^* \)-reachability graph of a protocol.

**Lemma 5.49**

Let \( G \) be a global state that is removed from the DFS stack during the construction of the \( \ell(J, K)^* \)-reachability graph, with \( G \equiv_{\{ r \mid J, K \}} H \) and \( \emptyset \neq \emptyset \), then there exist a sequence of leap sets \( \emptyset \) with \( \emptyset \downarrow \text{lin}(\emptyset) \), a transition sequence \( \emptyset \) and a global state \( H \uparrow \) such that \( G \equiv_{\{ r \mid J, K \}} H \uparrow \) and

i) \( \emptyset \equiv_{H_0} \emptyset \);

ii) \((s, y)\) is a ur-pair wrt \( J \) in \( H \uparrow \emptyset \downarrow \text{pref}(\emptyset) \): \( G \equiv_{\{ r \mid J, K \}} \emptyset \equiv_{\{ r \mid J, K \}} G \) and \((s, y)\) is a ur-pair in \( G \uparrow \)

iii) \((s, x)\) is a bo-pair wrt \( K \) in \( H \uparrow \emptyset \downarrow \text{pref}(\emptyset) \): \( G \equiv_{\{ r \mid J, K \}} \emptyset \equiv_{\{ r \mid J, K \}} G \) and \((s, x)\) is a bo-pair in \( G \uparrow \)

**Proof:** By repeated application of Lemma 5.48, akin to the proof of Lemma 5.38.

Lemma 5.49 is in effect identical to Lemma 5.38. Hence, the results stated in theorems 5.39-5.42
also hold for the $\ell_2(J, K)^*$-reachability relation (see the proofs of these theorems). We summarize with a corollary.

**Corollary 5.50**

For a protocol $\square$, indefinite progress, freedom of non-executable transitions, and freedom of unspecified receptions wrt $J$ and buffer overflows wrt $K$ are decidable if $L_2(J, K)_U^*$ is finite.  

**Example 5.51**

The $\ell_2^*$-reachability graph (i.e. $J = K = \square$) of the protocol of Example 5.6 is shown in Figure 5.8, as part of the $\ell^*$-reachability graph already given in Figure 5.5. The dashed nodes and edges indicate the $\ell^*$-reachable global states and transitions that are no longer explored, i.e. the $\ell_2^*$-reachability graph consists of only 9 nodes and 13 edges versus 10 nodes and 18 edges for the $\ell^*$-reachability graph. Observe that this difference is due in particular to the global states $G^0$ and $G^5$, where $xleap-2(G^k) \not\subseteq xleap(G^k)$ since the proper leap sets in these states do not close a cycle on the “current” DFS stack.

![Figure 5.8 The $\ell_2(\square, \square)^*$-reachability graph of the protocol of Example 5.6.](image-url)
5.6 Related work: LRA versus simultaneous reachability analysis

LRA borrows ideas from earlier work by Itoh & Ichikawa [II83] and by Özdemir & Ural [ÖU94, ÖU95, Özd95]. Their respective relief strategies also entail the concurrent execution of transitions at global states and tackle the issue of potentially executable transitions (cf. Section 4.1.2 and Section 5.1). Itoh & Ichikawa proposed a technique to explore only the reduced implementation sequences of a protocol, which constitute a subset of all the possible protocol executions (see Chapter 3). This technique is limited to the detection of non-progress states, however, and it imposes constraints on the structures of the processes in a protocol. All the processes are required to synchronize on their initial (process) states after a finite number of execution steps and no process is allowed to have a cyclic execution that does not pass through its initial state [II83]. The latter amounts to eluding any embedded cycles in the process graph of a process, which is clearly restrictive in practice for even a simple data transfer protocol usually exhibits such cycles (e.g. the retransmission part of the sender in an alternating bit protocol).

Özdemir and Ural generalized the idea of executing sets of concurrent transitions as a relief strategy for detecting all four types of logical errors, and without confining any of the protocol attributes [ÖU94, ÖU95, Özd95]. Their so-called simultaneous reachability analysis (SRA) thus applies to protocols in the CFSM model with an arbitrary number of processes, an arbitrary communication topology and arbitrary process structures. SRA is certainly the technique closest to LRA and, in fact, LRA has largely emerged as an incremental improvement of SRA [SU95b]. Similar to LRA, SRA governs the execution of leap sets in global states, called simultaneously executable sets in [ÖU95, Özd95], in order to detect non-progress states, non-executable transitions, unspecified receptions and buffer overflows in a protocol. This section gives an analytical comparison between the two techniques. An empirical comparison is included in Chapter 6. For ease of presentation we take LRA as defined in Section 5.1 through Section 5.4. It should be clear that the results established are then valid also for the refined, “depth-first search” version of LRA discussed in Section 5.5.

5.6.1 Detecting non-progress states and non-executable transitions

Unlike LRA, SRA does not support the option to carry out the detection of non-progress states separate from the detection of non-executable transitions. Where LRA employs the set \( \text{pleap}(G) \) in a global state \( G \) for detecting non-progress states alone (see Section 5.1), and the extended set \( \text{xpleap}(G) \) for detecting non-progress states and non-executable transitions (see Section 5.2), SRA always employs the set \( \text{sses}(G) \uplus \text{leap}(G) \) of so-called selected simultaneously executable sets for the detection of these two types of logical errors [ÖU95, Özd95].
**Definition 5.52**

Let $G$ be a global state of a protocol $\mathcal{G} = (\{P_i | i \in I\}, L)$. The set $sses(G)$ of *selected simultaneously executable sets* in $G$ is defined as follows:

$$sses(G) = \{ T | T \subseteq \text{leap}(G) \cap \text{act}(T) \supseteq \{ i \cap I | X_i(G) \neq \emptyset \cap P_i(G) = \emptyset \} \}$$

Informally, every selected simultaneously executable set in a global state $G$ obeys the following two rules: (1) it *must* contain an executable transition from each process with executable and no potentially executable transitions at $G$, and (2) it *may* contain an executable transition from any process with both executable and potentially executable transitions at $G$. The leap sets in $\text{pleap}(G)$ and $\text{xpleap}(G)$ also adhere to the first rule, but not to the second rule. Indeed, by definitions 5.4 and 5.16, every leap set in $\text{pleap}(G)$ or in $\text{xpleap}(G)$ contains *at most one* transition from among the processes with potentially executable transitions at $G$, whereas a leap set in $sses(G)$ can have multiple transitions from such processes. The difference between the three sets is further illustrated in Example 5.53, and expressed formally by Proposition 5.54.

**Example 5.53**

For the protocol of Example 5.6, $X(G^0) = \{ t_1^1, t_1^2, t_3^1, t_3^2 \}$ and $P(G^0) = \{ t_2^1, t_2^2 \}$. We already derived $\text{pleap}(G^0) = \{ \{ t_3^1, t_3^2 \} \}$ and $\text{xpleap}(G^0) = \{ \{ t_3^1, t_3^2 \}, \{ t_3^1, t_3^2 \}, \{ t_3^1, t_3^2 \}, \{ t_3^1, t_3^2 \} \}$, and now $sses(G^0) = \{ \{ t_3^1, t_3^2 \}, \{ t_3^1, t_3^2 \}, \{ t_3^1, t_3^2 \}, \{ t_3^1, t_3^2 \} \}$. The additional selected simultaneously executable set $\{ t_3^1, t_3^2 \}$ contains more than one transition from among the processes with potentially executable transitions at $G^0$.

**Proposition 5.54**

For a set of sets $S$, let $\text{min}(S) = \{ s | S \supseteq \bigcup s \supseteq S: s \subseteq S \}$, then

$$\text{pleap}(G) = \text{min}(sses(G)) \supseteq \text{xpleap}(G) \supseteq \text{sses}(G)$$

**Proof:** It is not difficult to see that $sses(G)$ can be defined equivalently in terms of the “wait-set” $\text{wait}(G)$ (see Definition 5.4) as follows:

$$sses(G) = \{ T | T \subseteq \text{leap}(G) \cap \text{act}(T) \supseteq \{ i \cap I | i \cap \text{wait}(G) \} \}$$

if $\text{wait}(G) \cap I$

$$sses(G) = \text{leap}(G)$$

otherwise.

The proposition is then immediate by the definitions of $\text{pleap}(G)$ and $\text{xpleap}(G)$ (Definition 5.4 and Definition 5.16, respectively).
As a direct consequence of Proposition 5.54, for any protocol both the $\ell$-reachability graph and the $\ell^*$-reachability graph are subgraphs of the corresponding “simultaneous reachability graph” resulting from SRA. LRA thus explores at most as many global states and transitions as SRA for detecting non-progress states and non-executable transitions. In general LRA can be expected to perform significantly better than SRA. This is evident especially for protocols with state spaces that manifest a wide distribution of potentially executable transitions. Consider for instance a global state $G$ where $k_1$ is the number of processes with executable transitions and without potentially executable transitions at $G$, and where $k_2$ is the number of processes with both executable and potentially executable transitions at $G$ (0 \[ k_1, k_2 \leq n \]). Assume for simplicity that all these processes have the same number $m$ of executable transitions at $G$. The cardinalities of $pleap(G)$, $xpleap(G)$ and $sses(G)$ are then as follows:

\[
\begin{align*}
|pleap(G)| &= m^{k_1} & \text{if } k_1 > 0 \text{ (i.e. } wait(G) \not\subseteq I) \\
|pleap(G)| &= k_2 \cdot m & \text{otherwise} \\
|xpleap(G)| &= |pleap(G)| + k_2 \cdot m & \text{if } k_1 > 0 \\
|xpleap(G)| &= |pleap(G)| & \text{otherwise} \\
|sses(G)| &= |pleap(G)| + |pleap(G)| \cdot \prod_{j=1}^{k_1} \left( \frac{k_2}{j} \right) m^j \\
&= |pleap(G)| + |pleap(G)| \cdot ((m+1)^{k_2}-1) & \text{if } k_1 > 0 \\
|sses(G)| &= |pleap(G)| + \prod_{j=2}^{k_1} \left( \frac{k_2}{j} \right) m^j \\
&= |pleap(G)| + ((m+1)^{k_2}-1-k_2 \cdot m) & \text{otherwise}
\end{align*}
\]

One can see that the size of $sses(G)$ grows very rapidly for increasing $k_2$: $|sses(G)| - |pleap(G)|$ is exponential in $k_2$ whereas $|xpleap(G)| - |pleap(G)|$ is only linear in $k_2$. Overall, SRA may thus compute and execute a significantly larger number of leap sets during state exploration than LRA. In the next chapter we will evaluate empirically the impact of this on the number of global states and transitions explored by SRA and LRA for detecting non-progress states and non-executable transitions, and on the actual space and time consumed by both techniques.

### 5.6.2 Detecting unspecified receptions

The detection of unspecified receptions by SRA proceeds in two stages. A given protocol is first augmented with extra receive transitions. These transitions are guaranteed to be non-executable and hence they do not alter the behavior of the protocol. State exploration is then carried out for the augmented protocol in the exact same way as described above, namely by executing selected simultaneously executable sets in global states [ÖU95, Özd95].
**Definition 5.55**
Let $\mathbf{I} = (\{P_i \mid i \in I\}, L)$ be a protocol, and denote by $@_{ij}$ a new unique message from process $P_i$ to process $P_j$ such that $@_{ij} \cup \bigcup_{i \in I} M_i$, for each $(i, j) \in L$. The triple $(s, +@_{ij}, s)$, with $s \in S_j$, is said to be an extraneous receive transition for $\mathbf{I}$ iff $M_{ij} \neq \emptyset$ and there exists no $(s_j, +y, s_k) \in \mathbf{I}$ such that $s_j = s$ and $y \in \mathbf{I}$. 

\[\text{for all } (i, j) \text{ in } J \text{ do}\\ \text{if } M_{ij} \neq \emptyset \text{ then}\\ \text{for all } s \text{ in } S_j \text{ do}\\ \text{if there is no receive transition at } s\\ \text{that involves a message from } M_{ij}\\ \text{then add } (s, @_{ij}, s) \text{ to } \mathbf{I}\\\]

**Figure 5.9** Constructing $\mathbf{I}$. 

A protocol $\mathbf{I}$ is augmented with respect to $J \in L$ by adding to $\mathbf{I}$ every extraneous receive transition $(s, +@_{ij}, s)$ for which $(i, j) \in J$. Denote the resulting protocol by $\mathbf{I}$. As illustrated in Figure 5.9, constructing $\mathbf{I}$ takes $O(\bigcup_{(i,j) \in J} |S_j|)$ time, where $|S_j|$ is the number of process states of process $P_j$. It is clear that all extraneous receive transitions are non-executable, since no matching send transitions are specified. Therefore, $R_{\mathbf{I}} = R_{\mathbf{I}}$ for any $J$. The application of SRA to $\mathbf{I}$ instead of $\mathbf{I}$ now reveals all unspecified receptions (ur-pairs) wrt $J$ in $\mathbf{I}$ [ÖU95, Özd95]. Recall from Section 5.4 that LRA uses the set $xpleap(G, J, \emptyset)$ for this purpose. In comparing the two techniques we arrive at the following proposition.

**Proposition 5.56**
Let $G$ be a global state of a protocol $\mathbf{I}$, and $G$ the corresponding global state of $\mathbf{I}$, then

\[\text{pleap}(G, J, \emptyset) = \text{min}(\text{sses}(G[\mathbf{I}]) \in \text{xpleap}(G, J, \emptyset) \in \text{sses}(G[\mathbf{I}])\]

**Proof:** Similar to the proof of Proposition 5.54, we show that $\text{sses}(G[\mathbf{I}])$ is defined in terms of the “wait-set” $\text{wait}(G, J, \emptyset)$ as follows:

\[\text{sses}(G[\mathbf{I}]) = \{ T \mid T \in \text{leap}(G) \in \text{act}(T) \supseteq \{ i \in I \mid i \in \text{wait}(G, J, \emptyset) \} \} \]

if $\text{wait}(G, J, \emptyset) \in I$

\[\text{sses}(G[\mathbf{I}]) = \text{leap}(G)\]

otherwise.

The inclusion $\text{xpleap}(G, J, \emptyset) \in \text{sses}(G[\mathbf{I}])$ is then immediate by definition of $\text{pleap}(G, J, \emptyset)$ and $\text{xpleap}(G, J, \emptyset)$ (see Definition 5.30 and Definition 5.31, respectively).
We need to prove that \( \{ i \ [ I \mid i \ [ wait(G, J, \emptyset) \} = \{ i \ [ I \mid X_i(G) \neq \emptyset \ [ P_i(G) = \emptyset \} \) (see Definition 5.52), or equivalently, that \( wait(G, J, \emptyset) = \{ i \ [ I \mid X_i(G) = \emptyset \ [ P_i(G) \neq \emptyset \) \), where \( wait(G, J, \emptyset) = \{ i \ [ I \mid X_i(G) = \emptyset \ [ P_i(G) \neq \emptyset \ [ (j, i) \ [ J: c_{ji}^G = \emptyset \) \) (see Definition 5.30). For this it is sufficient to show that the following two claims hold true:

i) \( X_i(G) = \emptyset \) iff \( X_i(G) = \emptyset \);

ii) \( P_i(G) \neq \emptyset \) iff \( P_i(G) \neq \emptyset \ [ (j, i) \ [ J: c_{ji}^G = \emptyset \) \)

Remark that \( G \) and \( G \) are the same global state (viz. with the same process states and the same channel contents), except for possible extraneous receive transitions defined at process states in \( G \) as a result of the augmentation. Since these transitions are non-executable it follows directly that \( X_i(G) = X_i(G) \), which proves claim (i). Regarding claim (ii), for the “only-if” part suppose that \( P_i(G) \neq \emptyset \) and let \( i \ [ P_i(G) \). It is clear that \( i \ [ P_i(G) \) if \( i \) is not an extraneous receive transition, while \( (j, i) \ [ J: c_{ji}^G = c_{ji}^G = \emptyset \) if \( j \) is an extraneous receive transition. For the “if” part, \( P_i(G) \neq \emptyset \) implies that \( P_i(G) \neq \emptyset \). The only case left is then that \( P_i(G) = \emptyset \ [ (j, i) \ [ J: c_{ji}^G = \emptyset \) In this case \( \emptyset \) has no receive transition at \( s_{ij}^G \) involving a message from \( M_{ij} \) because otherwise \( P_i(G) \neq \emptyset \). Hence, \( (s_{ij}^G, +@_{ji}, s_{ij}^G) \) is an extraneous receive transition for \( \emptyset \) and, moreover, this transition is potentially executable at \( G \) since \( c_{ji}^G = \emptyset \) Again, \( P_i(G) \neq \emptyset \).

As before, simple combinatorics testify that the difference in size between \( xpleap(G, J, \emptyset) \) and \( sses(G) \) can be substantial. We conclude from Proposition 5.56 that the \( \ell(J, \emptyset)^* \)-reachability graph obtained by LRA for the original protocol \( \emptyset \) is a subgraph of the simultaneous reachability graph obtained by SRA for the augmented protocol \( \emptyset \). Taking into account also the computational overhead associated with the augmentation procedure, SRA will thus utilize more space and time for detecting ur-pairs than LRA.

**Example 5.57**

The extraneous receive transitions for the protocol of Example 5.6 (see Figure 5.2) are:

- \( (21, +@_{12}, 21) \)
- \( (30, +@_{23}, 30) \)
- \( (30, +@_{43}, 30) \)
- \( (22, +@_{12}, 22) \)
- \( (31, +@_{23}, 31) \)
- \( (40, +@_{34}, 40) \)

Augmenting this protocol wrt \( J = \{ (2, 3), (4, 3) \) \) is accomplished by adding the three transitions \( (30, +@_{23}, 30) \), \( (31, +@_{23}, 31) \) and \( (30, +@_{43}, 30) \). The application of SRA to the augmented protocol then yields a simultaneous reachability graph consisting of 25 nodes and 84 edges. In contrast, the \( \ell(J, \emptyset)^* \)-reachability graph of the original protocol consists of 22 nodes and 52 edges (see Example 5.44).
5.6.3 Detecting buffer overflows

Before comparing SRA and LRA for the detection of buffer overflows (bo-pairs), we should first point out that the objectives of both techniques are actually different. While LRA aims at detecting all bo-pairs that cause a channel overflow, SRA aims at identifying just the overflowed channels [ÖU95, Özd95]. For the latter it suffices to detect only one bo-pair (if any) per channel. Although not explicitly stated in [ÖU95, Özd95], we do recognize that SRA lends itself also for detecting all the bo-pairs in a protocol. Further left unmentioned in [ÖU95, Özd95] is the fact that this can be done for multiple channels at once, i.e. in a single verification run. It is stated instead that the detection of all overflowed channels necessitates the application of SRA once for every channel in the protocol. Nevertheless, when we take the same general objective of detecting all bo-pairs with respect to an index set \( K \) of channels (cf. Section 5.4) for both LRA and SRA, the argument in favor of LRA is once again that the \( \ell(\emptyset, K)^+ \)-reachability graph of any given protocol is guaranteed to be a subgraph of the simultaneous reachability graph resulting from SRA for detecting these bo-pairs. In the rest of this section we substantiate also this claim.

In order to determine the possibility of an overflow for a specific channel \( C_{jk} \), SRA employs the following subset of leap sets in a global state \( G \) [ÖU95, Özd95] (cf. Definition 5.52):

\[
\{T \not\subseteq \text{leap}(G) \mid \text{act}(T) \supseteq \{i \not\subseteq I \mid X_i(G) \neq \emptyset \not\subseteq P_i(G) = \emptyset \not\subseteq (i = k) \not\subseteq c_{jk}^G = \emptyset\}\}
\]

Accordingly, we provide a more general definition to deal with a set \( K \) of channels instead of just a single channel.

**Definition 5.58**

Let \( G \) be a global state of a protocol \( \not\subseteq = (\{P_i \mid i \not\subseteq I\}, L) \) and \( K \not\subseteq L \). Define the set \( sses(G, K) \) of selected simultaneously executable sets wrt \( K \) in \( G \) as follows:

\[
sses(G, K) = \{T \not\subseteq \text{leap}(G) \mid \text{act}(T) \supseteq \{i \not\subseteq I \mid X_i(G) \neq \emptyset \not\subseteq P_i(G) = \emptyset \not\subseteq (j, i) \not\subseteq K: c_{ji}^G = \emptyset\}\}
\]

We establish that \( xpleap(G, \emptyset, K) \) is included in \( sses(G, K) \), for any \( K \), and this proves that LRA outperforms SRA for the detection of bo-pairs.

**Proposition 5.59**

\( xpleap(G, \emptyset, K) \not\subseteq sses(G, K) \)

**Proof:** Notice that \( \{i \not\subseteq I \mid X_i(G) \neq \emptyset \not\subseteq P_i(G) = \emptyset \not\subseteq (j, i) \not\subseteq K: c_{ji}^G = \emptyset\not\subseteq I \not\subseteq wait(G, \emptyset, K) = \{i \not\subseteq I \mid X_i(G) \neq \emptyset \not\subseteq P_i(G) = \emptyset \not\subseteq (j, i) \not\subseteq K\} \) (see Definition 5.30), in particular because \( X_{ij}^*(G) = \emptyset \) if \( c_{ji}^G = \emptyset \). The set \( sses(G, K) \) can hence be defined equivalently as follows:
\[ \text{sses}(G, K) = \{ T \mid T \not\subset \text{leap}(G) \not\subset \text{act}(T) \supseteq \{ i \not\subset I \mid i \not\subset \text{wait}(G, \emptyset, K) \not\subset (j, i) \subset K: c_{ji}^j = \emptyset \} \} \]

\[ \text{sses}(G, K) = \text{leap}(G) \]

otherwise.

The inclusion \( xpleap(G, \emptyset, K) \not\subset \text{sses}(G, K) \) follows then readily by definition of \( pleap(G, \emptyset, K) \) and \( xpleap(G, \emptyset, K) \).

\[ \square \]

### 5.7 Summary

In this chapter we have developed a relief strategy, named *leaping reachability analysis* (LRA), for the verification of logical correctness properties of protocols defined in the CFSM model. We proved that, for any given protocol in this model, LRA maintains the power of conventional reachability analysis to detect all non-progress states (including deadlocks), all non-executable transitions, all unspecified receptions and all buffer overflows. Yet, in contrast to conventional reachability analysis where transitions are executed one at a time, LRA employs the *concurrent* execution of transitions at global states. By executing concurrent transitions collectively as sets, it “leaps” through the state space of a protocol and may thereby reduce significantly the number of global states and transitions explored. The potential impact of LRA is thus a large decrease in memory and time needed for verifying logical correctness properties.

LRA has been inspired to a large extent by earlier work of Özdemir & Ural on *simultaneous reachability analysis* (SRA) [ÖU94, ÖU95, Özd95]. In fact, we propose LRA as an incremental improvement of SRA. SRA similarly employs the execution of sets of concurrent transitions to verify the same four logical correctness properties, and is arguably the first relief strategy applicable to protocols in the CFSM model without confining any of the protocol attributes (viz. the number of processes in a protocol, its communication topology and the individual process structures). Through an analytical comparison we have shown that, for any protocol, the “reduced” reachability graph resulting from LRA is a subgraph of the one resulting from SRA, for each of the four logical correctness properties. Thus, LRA never explores more global states or transitions than SRA. Moreover, LRA never incurs more run-time overhead, and even eliminates the need for a protocol augmentation in detecting unspecified receptions. Using LRA instead of SRA is therefore a *no-risk improvement*. In the next chapter we will complement the analytical results with an empirical evaluation of the performance of LRA, with respect to SRA and conventional reachability analysis.