1 **function** $II(M = \langle S, s_I, T\rangle, G, opt, \epsilon)$
  // Preprocessing
2 **if** $opt = \max$ **then** $M :=$ `CollapseMECs`(M, G) *// collapse MECs*
3 $S_0 :=$ `Prob0`(M, G, opt), $S_1 :=$ `Prob1`(M, G, opt) *// identify 0/1 states*
4 $l := \{\, s \mapsto 0 \mid s \in S \setminus S_1 \,\} \cup \{\, s \mapsto 1 \mid s \in S_1 \,\}$ *// initialise lower vector*
5 $u := \{\, s \mapsto 0 \mid s \in S_0 \,\} \cup \{\, s \mapsto 1 \mid s \in S \setminus S_0 \,\}$ *// initialise upper vector*
  // Iteration
6 **while** $(u(s_I) - l(s_I))/l(s_I) > \epsilon$ **do** *// while relative error $> \epsilon$:*
7   **foreach** $s \in S \setminus (S_0 \cup S_1)$ **do** *// update non-0/1 states:*
8     $l(s) := opt_{\mu \in T(s)} \sum_{s' \in spt(\mu)} \mu(s') \cdot l(s')$ *// iterate lower vector*
9     $u(s) := opt_{\mu \in T(s)} \sum_{s' \in spt(\mu)} \mu(s') \cdot u(s')$ *// iterate upper vector*
10 **return** $\frac{1}{2}(u(s_I) - l(s_I))$

Alg. 1: Interval iteration for probabilistic reachability

steps require no changes as they are purely graph-based. The changes to the iteration part of the algorithm are straightforward: In line 6,

$$\textbf{while } (u(s_I) - l(s_I))/l(s_I) > \epsilon \textbf{ do } \ldots,$$

we round the results of the subtraction and of the division towards $+\infty$ to avoid stopping too early. In line 8,

$$l(s) := opt_{\mu \in T(s)} \textstyle\sum_{s' \in spt(\mu)} \mu(s') \cdot l(s'),$$

the multiplications and additions round towards $-\infty$ while the corresponding operations on the upper bound in line 9 round towards $+\infty$. Recall that all probabilities in the MDP are rational numbers, i.e. representable as $\frac{num}{den}$ with $num, den \in \mathbb{N}$. We assume that num and den can be represented exactly in the implementation. Then, in line 8, we calculate the floating-point values for the $\mu(s') = num/den$ by rounding towards $-\infty$. In line 9, we round the result of the corresponding division towards $+\infty$. Finally, instead of returning the middle of the interval in line 10, we return $[l(s_I), u(s_I)]$ so as not to lose any information (e.g. in case the result is compared to a constant as in the example of Sect. 3.1).

To ensure termination, we thus need to make one further change to the II of Alg. 1: In each iteration of the **while** loop, we additionally keep track of whether any of the updates to l and u changes the previous value. If not, we end the loop and return the current interval, which will be wider than the requested ϵ relative difference. We refer to II with all of the these modifications as *safely rounding interleaved II* (SR-III) in the remainder of this paper.

function $\mathit{SR\text{-}SII}(M = \langle S, s_I, T\rangle, G, opt, \epsilon)$

  ... (*preprocessing as in Alg. 1*) ...

  repeat

    $chg := \mathit{false}$

    $\texttt{fesetround}(\mathit{towards}\ -\infty)$

    foreach $s \in S \setminus (S_0 \cup S_1)$ **do**

      $l_{new} := opt_{\mu \in T(s)} \sum_{s' \in spt(\mu)} \mu(s') \cdot l(s')$ // *iterate lower vector*

      if $l_{new} \neq l(s)$ **then** $chg := \mathit{true}$

      $l(s) := l_{new}$

    $\texttt{fesetround}(\mathit{towards}\ +\infty)$

    foreach $s \in S \setminus (S_0 \cup S_1)$ **do**

      $u_{new} := opt_{\mu \in T(s)} \sum_{s' \in spt(\mu)} \mu(s') \cdot u(s')$ // *iterate upper vector*

      if $u_{new} \neq u(s)$ **then** $chg := \mathit{true}$

      $u(s) := u_{new}$

  until $\neg chg \vee (u(s_I) - l(s_I))/l(s_I) \leq \epsilon$

  return $[l(s_I), l(s_I)]$

Alg. 2: Safely rounding sequential interval iteration (SR-SII) for x87 or SSE