Assume the generalization procedure unfolds this execution tree:

```
let rec generalize (already : ItemSet.t) (others : ItemSet.t) : unfolding =
  match ItemSet.min_elt_opt others with
  | None -> Return already
  | Some elt ->
     let others' = ItemSet.remove elt others in
     IfThenElse ((ItemSet.union already others'),
                 generalize already others',
                 generalize (ItemSet.add elt already) others');;
```

IfThenElse(s, t, f) queries if the set s is suitable and executes t or f accordingly. Whether the then branch is taken when s is queried is a Bernoulli random variable a_s.

P is a monotone function from the subsets of X to $\{0,1\}$, 0 identified with false, 1 with true. eval_test s picks a choice according to a_s, and eval_result s checks if $P(s)$. The probability that the execution tree produced by the above procedure returns an incorrect result ($\neg P(s)$) is:

```
let rec eval_probabilistic  (eval_test : ItemSet.t -> float)
          (eval_result : ItemSet.t -> float) = function
  | Return is -> eval_result is
  | IfThenElse(is, unf_t, unf_f) ->
     let p_true = eval_test is in
        p_true *. (eval_probabilistic eval_test eval_result unf_t)
        +. (1. -. p_true) *. (eval_probabilistic eval_test eval_result unf_f)
```

Now we try to maximize this probability over all P. For the sake of simplicity, assume temporarily that there are only two probability distributions for a_s depending on whether $P(s)$ is true or false. Then there are only two coefficients mistake_correct ($\P a_s = 0$ when $P(s)$) and mistake_incorrect ($\P a_s = 1$ when $\neg P(s)$), also denoted by e. At each step, we choose between these two distributions, but once the distribution for $\neg P(s)$ is picked then it must be picked for all the subtree, by monotonicity. Furthermore, once the algorithm believes that a set is admissible whereas it is not, the only possible outcome of branches in the execution tree is a wrong answer. We obtain:

```
let rec bound_probabilistic = function
  | Return is -> 0.0
  | IfThenElse(is, unf_t, unf_f) ->
     Float.max
       (mistake_incorrect +.
          (1. -. mistake_incorrect) *.  (bound_probabilistic unf_f))
       ((mistake_correct *. (bound_probabilistic unf_f)) +.
        ((1. -. mistake_correct) *. (bound_probabilistic unf_t)));;
```

Now note that this function no longer depends on the actual sets, but only on the structure of the tree, which depends only on the cardinal of the others set. We would obtain the same with calling this function with argument $|X|$:

```
let rec upper_bound_probabilistic = function
  | 0 -> 0.0
  | n ->
  let next_bound = upper_bound_probabilistic (n-1) in
  Float.max
     (mistake_incorrect +.
        (1. -. mistake_incorrect) *. next_bound)
     ((mistake_correct *. next_bound) +.
        ((1. -. mistake_correct) *. next_bound));;
```

The second argument to max simplifies to next_bound. Since next_bound $\leq$ 1, the first argument is always greater than or equal to the second, which simplifies into:

```
let rec upper_bound_probabilistic = function
  | 0 -> 0.0
  | n ->
     let next_bound = upper_bound_probabilistic (n-1) in
     (mistake_incorrect +.
          (1. -. mistake_incorrect) *. next_bound);;
```

In other words, $p_0 = 0$, $p_{n+1} = e + (1 - e)p_n$, and the closed form is $p_n = 1 - (1 - e)^n$.

Now in the above reasoning, allow probability distributions for a_s to vary across calls, but we now that $\mathbb{P}(a_s = 1)$ may differ across several s where $\neg P(s)$. We however know that $\mathbb{P}(a_s = 1) \leq \epsilon$ when $\neg P(s)$. With the same reasoning. Then the overall probability that the algorithm produces a wrong answer is bounded by

$$1 - (1 - \epsilon)^{|X|} \tag{1}$$

This bound is tight: it is reached when the only admissible set is X itself ($P(X)$ but $\neg P(X')$ for $X' \subsetneq X$) and $\mathbb{P}(a_s = 1) = \epsilon$ for all $X' \subsetneq X$.