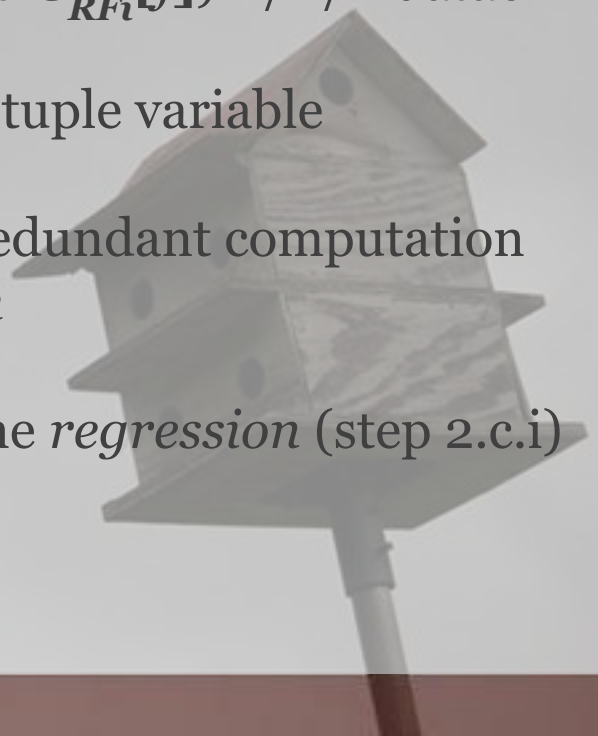
Finding Interpolant as Broad Race-free Condition

- *raceCons*(race constraint) = *paralCond*(parallel condition) /\ *pathCond*(path condition) /\ *raceCond*(race condition)
- *raceFreeCond* (race free variable condition):
  - e.g. `size=5 /\ chunk=2 /\ tM=2`
  - Initially *raceFreeCond* = `false`
- **Race-free** property: *raceFreeCond* /\ *raceCons* → UNSAT
- *itpl*: Craig interpolant
  - *raceFreeCond* /\ *raceCons* → UNSAT s.t.
    - *raceFreeCond* → *itpl*
    - *raceCons* /\ *itpl* → UNSAT
    - ***var***(*itpl*) = ***var***(*raceFreeCond*) ∩ ***var***(*raceCons*)

The Algorithm, in Readable Text

1. *Initializing: raceFreeCond* := False (***RaceFree*** := {False})
2. *Negative learning with case assignment*: *raceFreeConsUnexpl*(*raceFreeCond*) := *raceFreeCons* /\ ~*raceFreeCond*
   a. if SAT => goto step 3 with $vars_{RF}$:= SAT model
   b. if UNSAT => *return raceFreeCond* as *maximal race free condition*
   c. if unknown/timed-out
      i. *doing regression:* goto step 2 with *raceFreeConsUnexpl*($raceFreeCond_{regr}$), where $raceFreeCond_{regr}$ temporarily combinatorially hides some known race-free cases, including False, from *raceFreeCond*
      ii. *doing more existential quantification:* goto step 2 with ***var(****raceFreeCond****)*** := ***var(****raceFreeCond****)*** $\cup$ $\{v | v \in$ ***var(****raceCons****)*** /\ $v \notin$ ***var(****raceFreeCond****)***$\}$
      iii. *return raceFreeCond* as *best-known race free condition* if existential quantification upper bound is reached
3. *Case expansion: raceFreeCond* := *raceFreeCond* \/ $vars_{RF}$

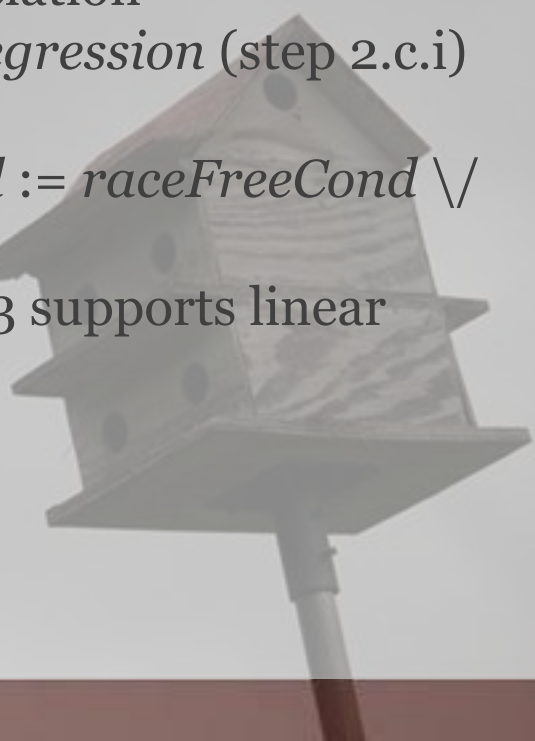
4. *Interpolation: itpl* := *interpolant*(*raceFreeCond*, *raceCons*)
   a. if SAT/unknown/timed-out/*itpl* = *raceFreeCond* =>
      $itpl_{turbo\_heu}$:= *interpolant*($raceFreeCond_{turbo\_heu}$, *raceCons*), where $raceFreeCond_{turbo\_heu}$:=
      i. *one without totally conflicting inv:* if $raceFreeCond_{turbo\_heu}$ doesn't *totally* conflict *inv*, goto step 4.c with *raceFreeCond* := $raceFreeCond_{turbo\_heu}$ \/ $itpl_{turbo\_heu}$ or $raceFreeCond_{turbo\_heu}$ (if $itpl_{turbo\_heu}$= $raceFreeCond_{turbo\_heu}$)
      ii. *subsumption learning: raceFreeCond* \/ *subsume_heu*(*raceFreeCond*), where *subsume_heu*(*raceFreeCond*) = /\ *name*($\boldsymbol{vars}_{RFi}[j]$) </=/> *value* ($\boldsymbol{vars}_{RFi}[j]$) (subsumption):
      While there's any single, double, triple, quad, ..., *n*-tuple variable combination $\boldsymbol{vars}_{RFi}$,
         1. bypassing (and caching) combinatorially redundant computation
         2. *analogical variable bounded subsumption*
         3. *simple bounded subsumption*
         4. including (unhiding) the cases hidden in the *regression* (step 2.c.i) and computing again



iii. *simple bounded linear learning: raceFreeCond* \/ *linear_heu*(*raceFreeCond*), where *linear_heu*(*raceFreeCond*) = (boundary) /\ linear relation:
While there's any pair of single, double, triple, quad, ..., *n*-tuple variable vector $\boldsymbol{vars\_pair}_{RFi}$,

1. eliminating shared variables
2. *computing linear relations* ($\boldsymbol{vars\_pair}_{RFi}[1] = a^{*}\boldsymbol{vars\_pair}_{RFi}[2] + b$) for each pair of the rest variable vectors
3. *combinatorially bounding all non-eliminated variables* if there's no interpolant for the heuristics with just linear relation
4. including (unhiding) the cases hidden in the *regression* (step 2.c.i) and computing again

b. else (if UNSAT and *itpl* ≠ *raceFreeCond*) => *raceFreeCond* := *raceFreeCond* \/ *itpl*,
c. double-check *raceFreeCond* /\ *raceCons* by Z3 (for that iZ3 supports linear integer algorithm only)
   i. should be UNSAT => goto step 2,
   ii. if SAT/unknown/timed-out => warning buggy iZ3



Main Algorithm Flowchart

From race-free constraint unexplored unknown/timed-out
or interpolation
or dead-end breakthrough - subsumption/linear learning
flowchart

1. Initializing
raceFreeCond := False
(***RaceFree*** := {False})

2. Negative learning with case assignment
raceFreeConsUnexpl(*raceFreeCond*) :
= raceFreeCons /\ ~*raceFreeCond*

2.c. Unknown/timed-out
To race-free constraint unexplored unknown/timed-out flowchart

2.a. SAT
$vars_{RF}$:= SAT model

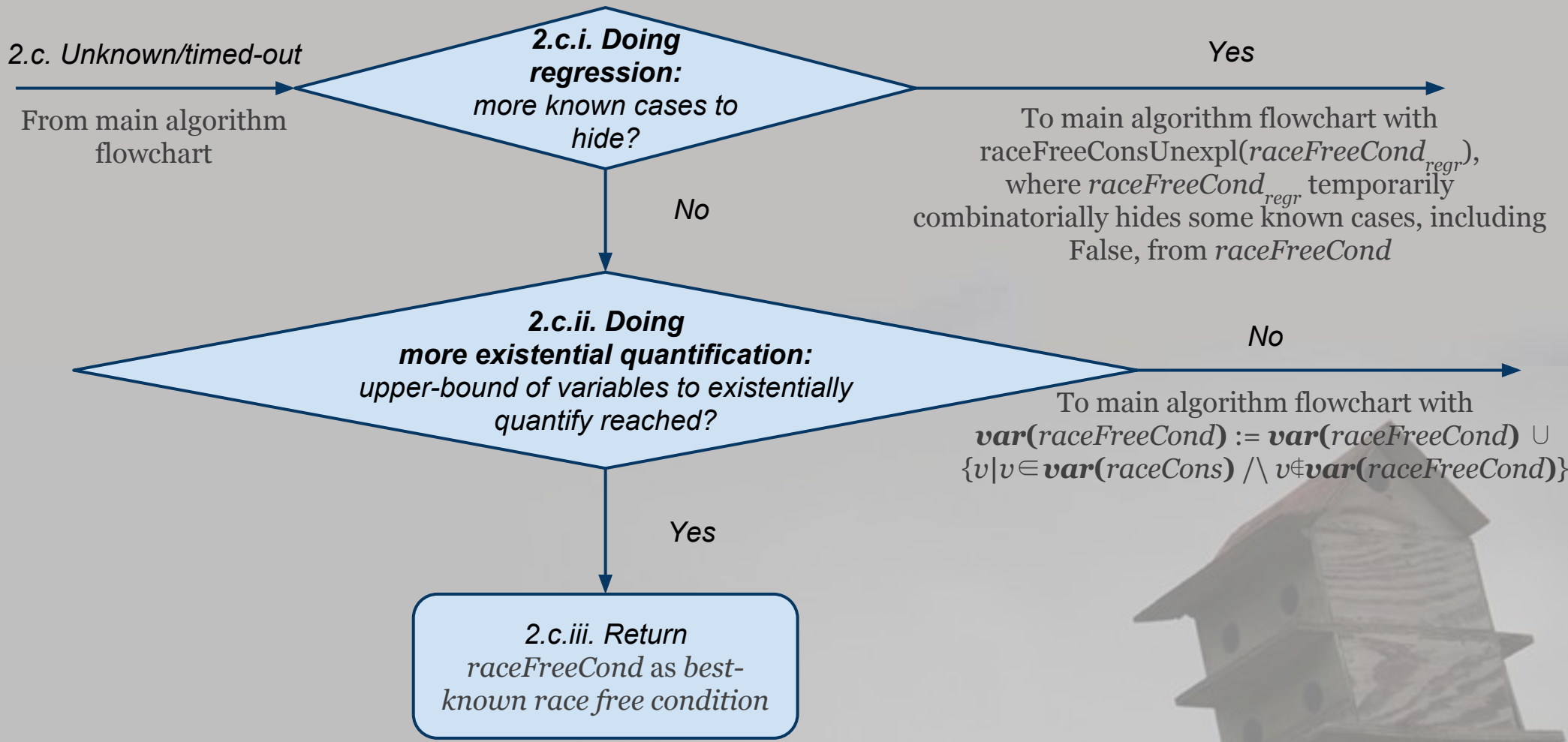
2.b. UNSAT

3. Case expansion
raceFreeCond := *raceFreeCond* \/ $vars_{RF}$

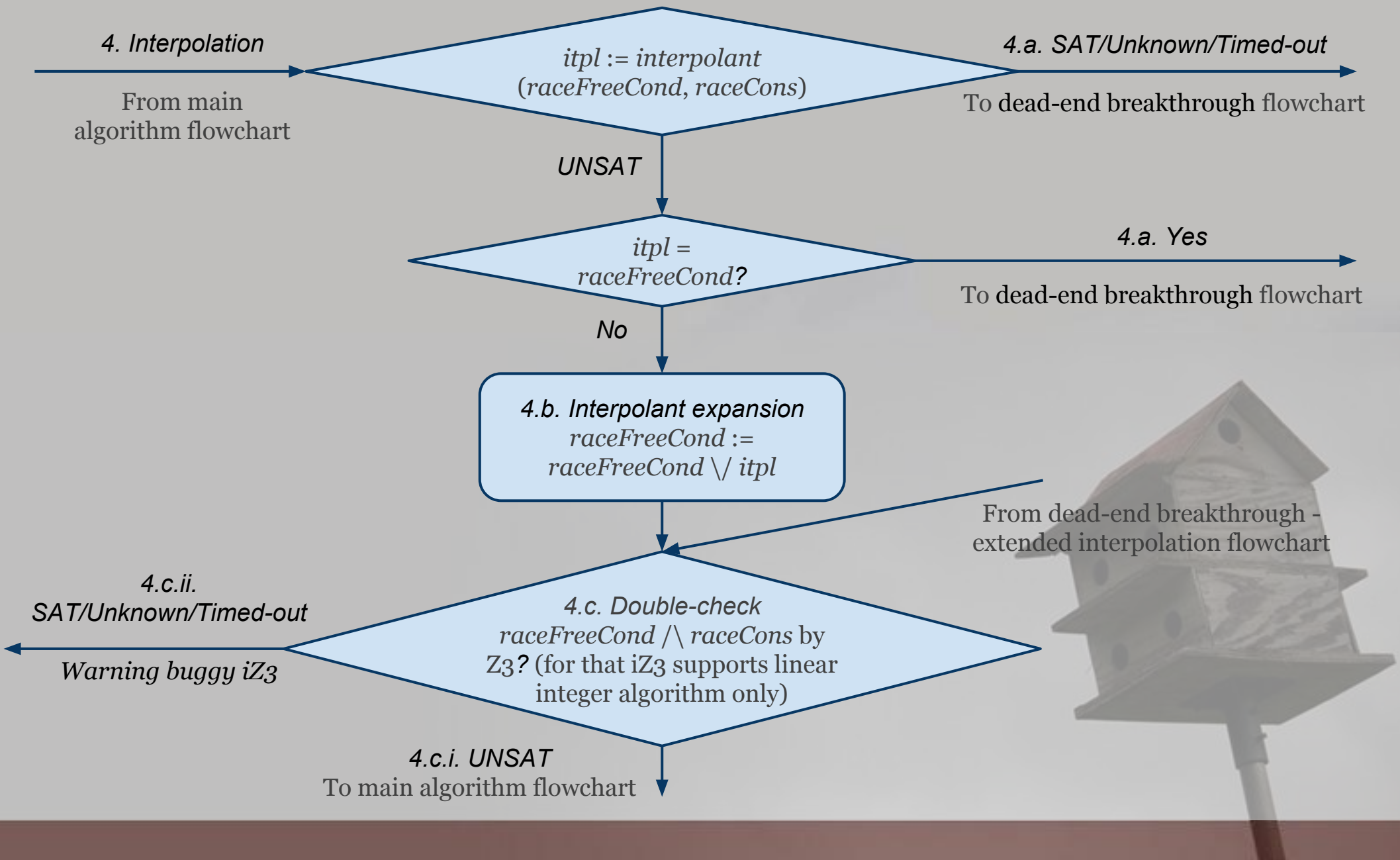
Return
raceFreeCond as *maximal race free condition*

4. Interpolation
To interpolation flowchart

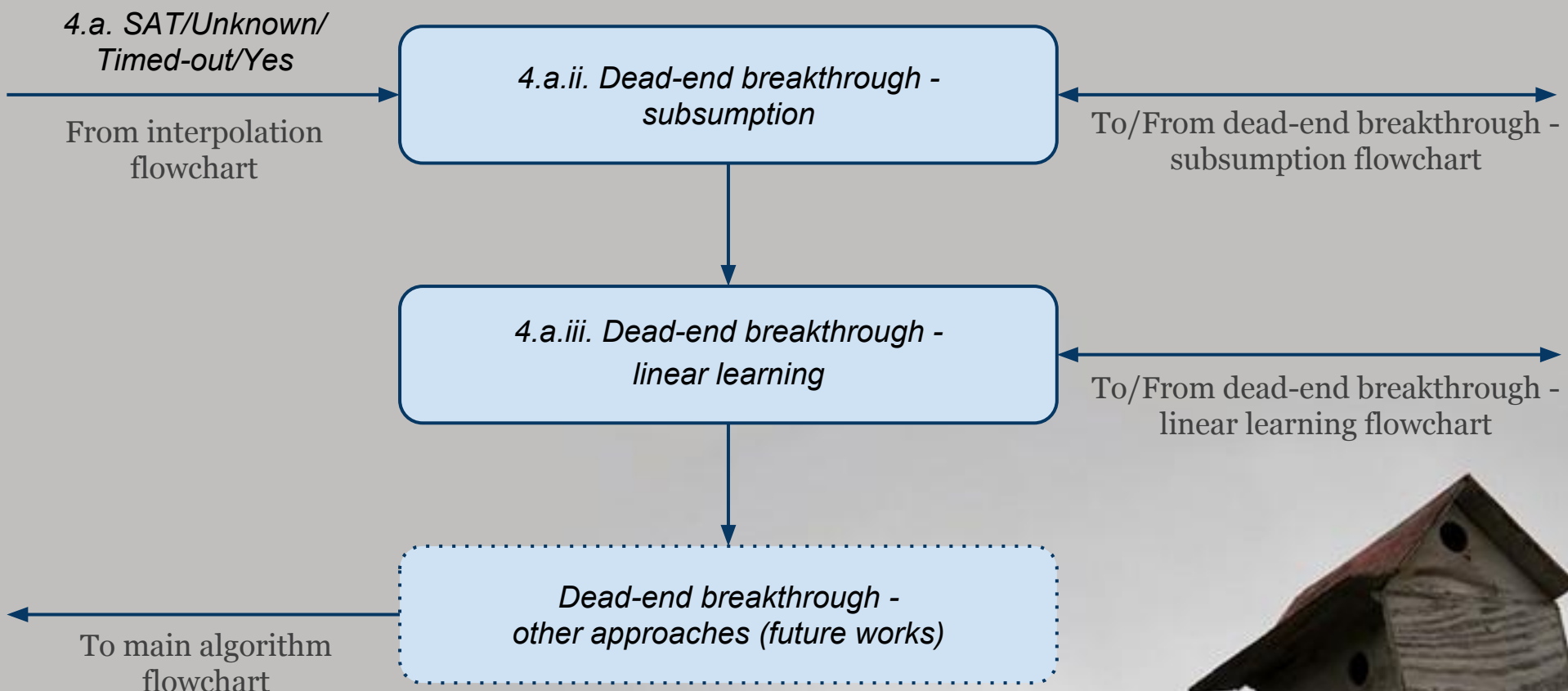
Race-free Constraint Unexplored Unknown/Timed-out Flowchart



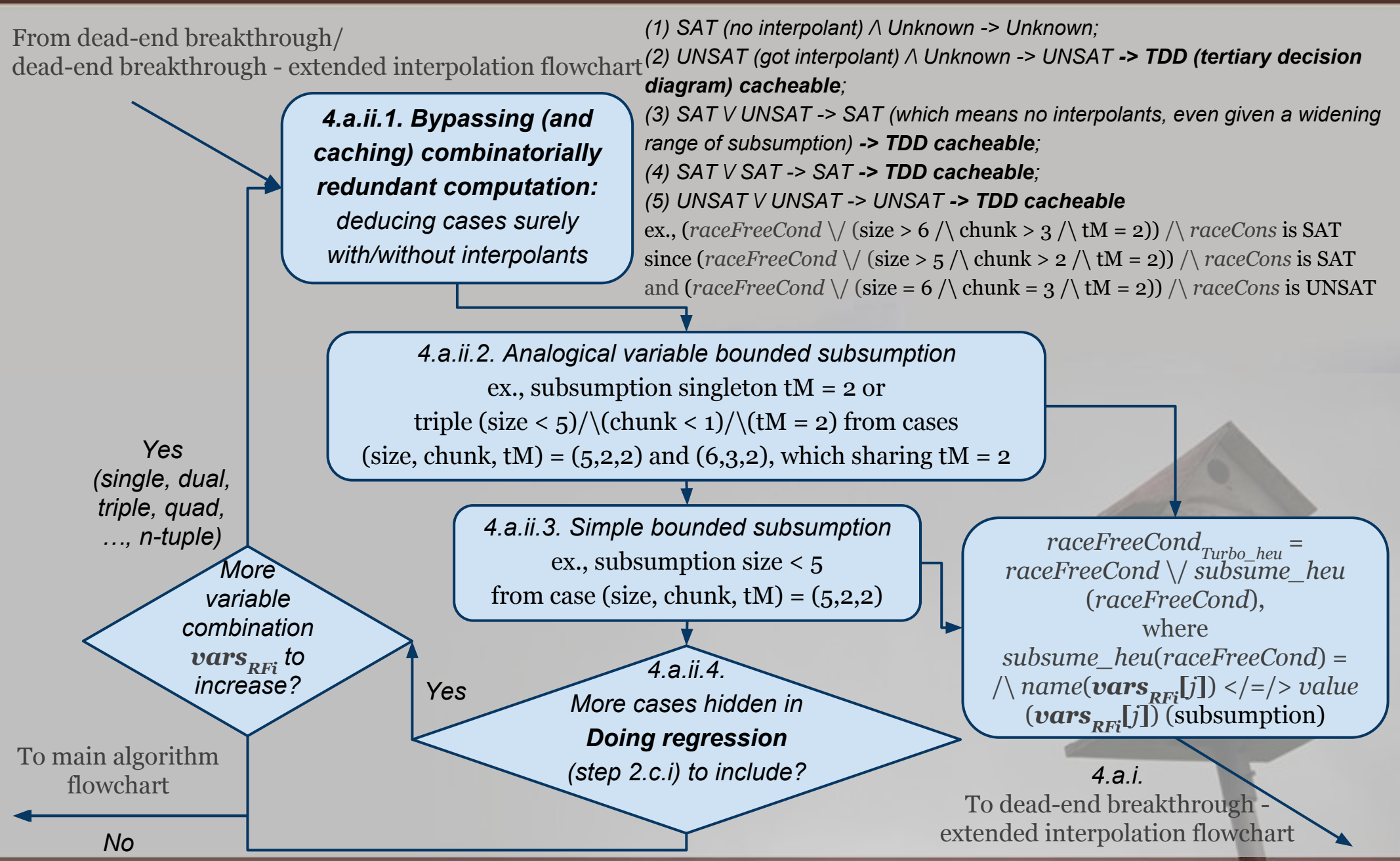
Interpolation Flowchart



Dead-end Breakthrough Flowchart



Dead-end Breakthrough - Subsumption Flowchart



Dead-end Breakthrough - Linear Learning Flowchart

From dead-end breakthrough flowchart

Simple bounded linear learning:

4.a.iii.1. eliminating shared variables

-> 4.a.iii.2. computing linear relations

$(\boldsymbol{vars\_pair}_{RFi}[1] = a^{*}\boldsymbol{vars\_pair}_{RFi}[2] + b)$

for each pair of the rest variable vectors

ex., size = chunk + 3

from cases (size, chunk, tM) = (5,2,2) and (6,3,2)

4.a.iii.3. Combinatorially bounding all non-eliminated variables if returned without any interpolant

$raceFreeCond_{turbo\_heu} = raceFreeCond \setminus/ linear\_heu(raceFreeCond)$, where

$linear\_heu(raceFreeCond)$ = (boundary) /\ linear relation

4.a.i.
To/From dead-end breakthrough - extended interpolation flowchart

Any more pair of variable vector $\boldsymbol{vars\_pair}_{RFi}$ *?*

Yes

(single, dual, triple, quad, …, n-tuple)

Yes

No

4.a.iii.4. More cases hidden in ***Doing regression*** *(step 2.c.i) to include ?*

No

To main algorithm flowchart

Dead-end Breakthrough - Extended Interpolation Flowchart