CS 480

Introduction to Artificial Intelligence

January 30, 2024

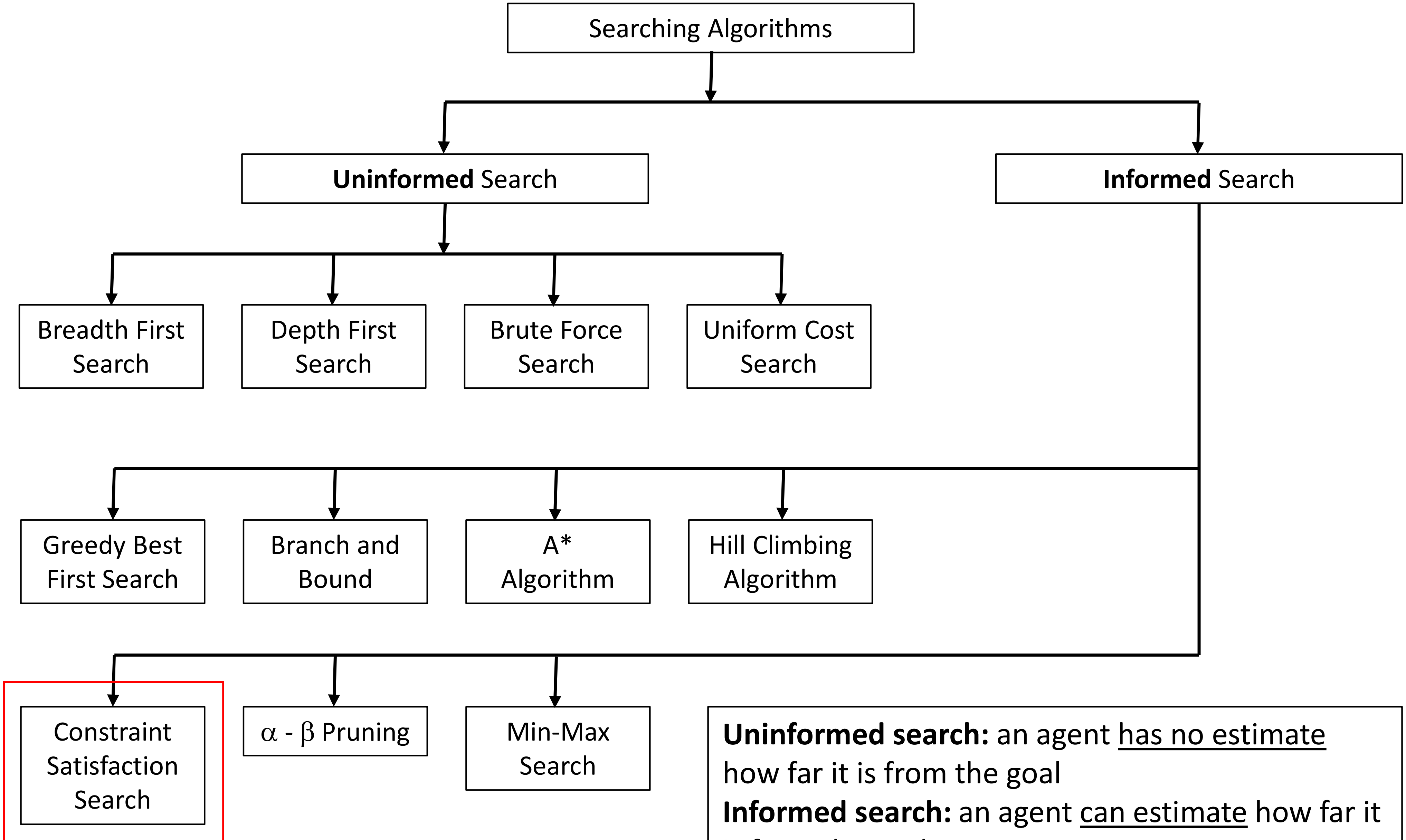
Announcements / Reminders

- Please follow the Week 03 To Do List instructions (if you haven't already):
- Quiz #02: due on Sunday (02/04/24) at 11:59 PM CST
  - New quiz will be posted on Monday!
- Written Assignment #01 due on Tuesday (02/06/24) at 11:59 PM CST
- Programming Assignment #01 due on Sunday (02/18/24) at 11:59 PM CST

Plan for Today

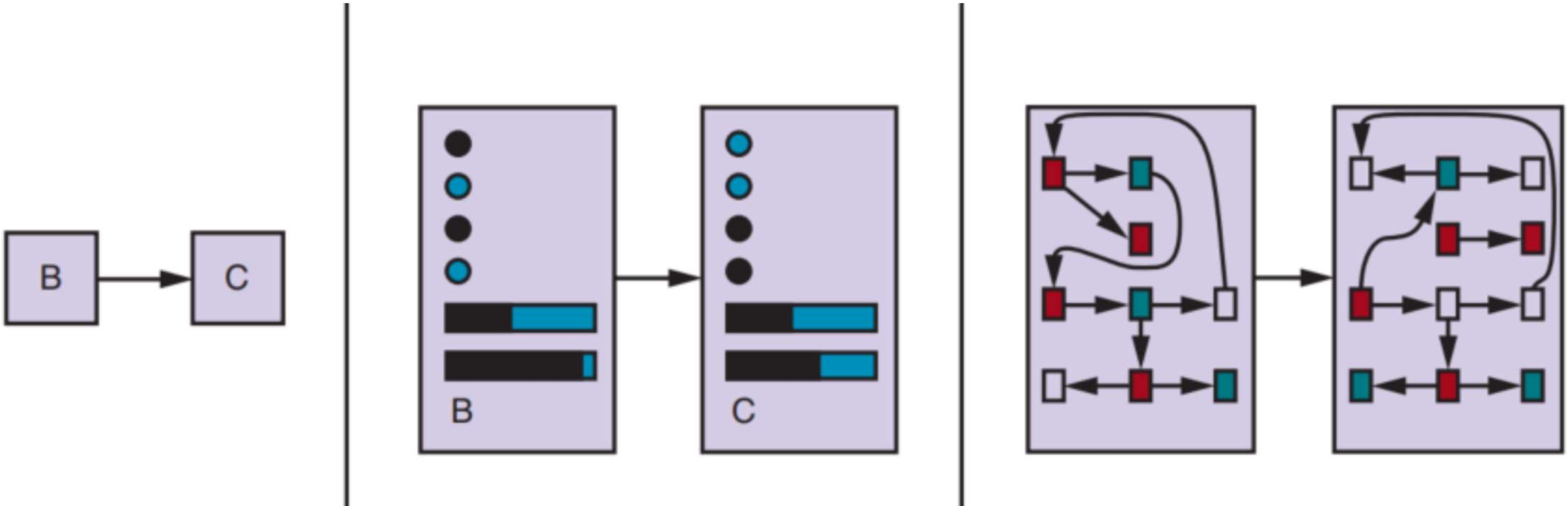
- **Constraint Satisfaction Problems (CSPs)**

Selected Searching Algorithms

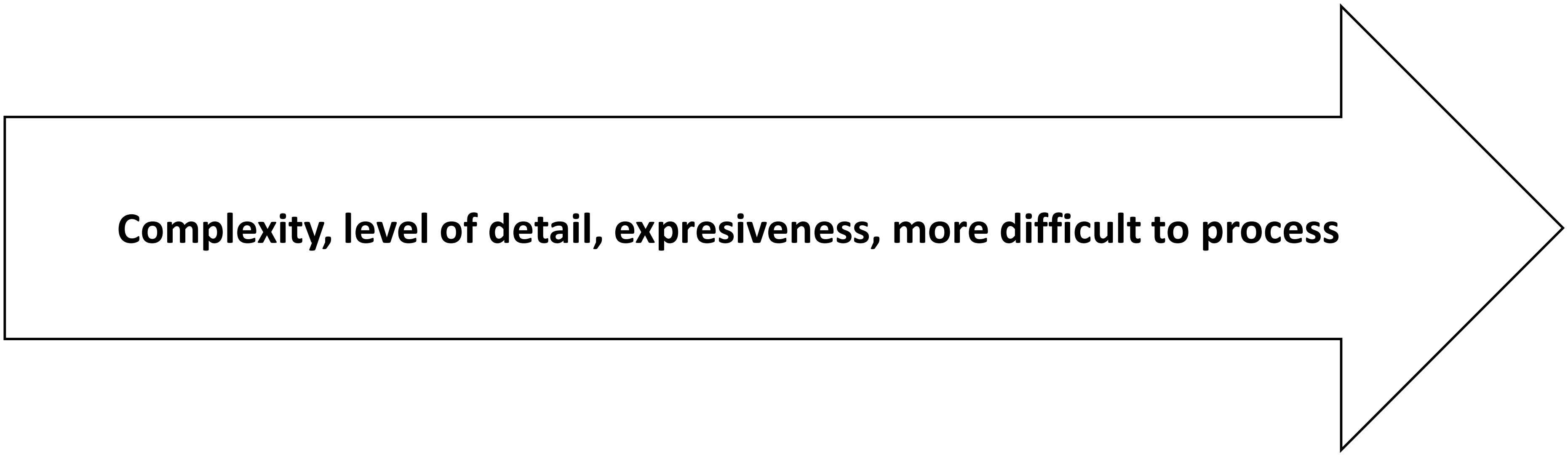


Uninformed search: an agent has no estimate how far it is from the goal
Informed search: an agent can estimate how far it is from the goal

State and Transition Representations



(a) Atomic (b) Factored (c) Structured



Constraint Satisfaction Problem

A Constraint Satisfaction Problem (CSP) consists of three components:

- **a set of variables** $X = \{X_1, ..., X_n\}$
- **a set of domains** $D = \{D_1, ..., D_n\}$
- **a set of constraints** C **that specify allowable combinations of values**

- **A domain** D_i **is a set of allowable values** $\{v1, ..., vk\}$ **for variable** X_i
- **A constraint** C_j **is a** $\langle \text{scope, relation} \rangle$ **pair, for example** $\langle (X1, X2), X1 > X2 \rangle$

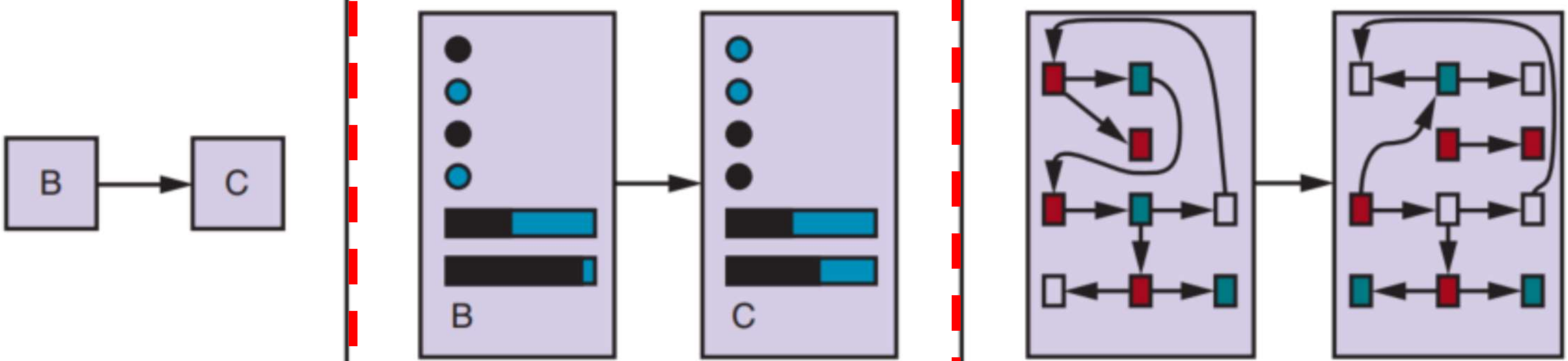
Constraint Satisfaction Problem

The goal is to find an assignment (variable = value):

$$\{X_1 = v_1, \ldots, X_n = v_n\}$$

- **If NO constraints violated: consistent assignment**
- **If ALL variables have a value: complete assignment**
- **If SOME variables have NO value: partial assignment**
- **SOLUTION: consistent and complete assignment**
- **PARTIAL SOLUTION: consistent and partial assignment**

State Representations



(a) Atomic (b) Factored (c) Structured

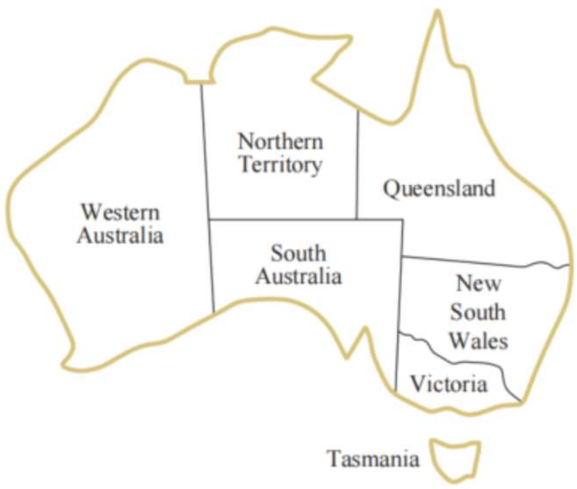
- Searching
- Hidden Markov models
- Markov decision process
- Finite state machines

- Constraint satisfaction algorithms
- Propositional logic
- Planning
- Bayesian algorithms
- Some machine learning algorithms

- Relational database algorithms
- First-order logic
- First-order probability models
- Natural language understanding (some)

CSP Example: Map Coloring

Problem:



Color this map in a way that no two neighbors have same color

Variables:

X = {WA, NT, Q, NSW, V, SA, T}

Variable Domains:

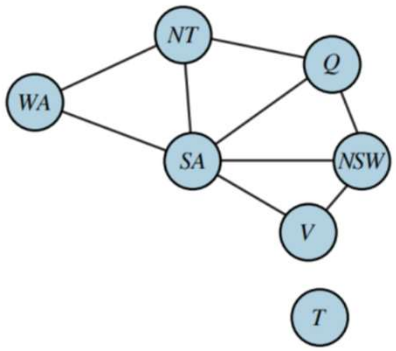
D_{WA} = {RED, GREEN, BLUE}
D_{NT} = {RED, GREEN, BLUE}
D_{Q} = {RED, GREEN, BLUE}
D_{NSW} = {RED, GREEN, BLUE}
D_{V} = {RED, GREEN, BLUE}
D_{SA} = {RED, GREEN, BLUE}
D_{T} = {RED, GREEN, BLUE}

Constraints (Rules):

- **Neighboring regions have to have DISTINCT colors:**

CONSTRAINTS = C = {SA $\neq$ WA, SA $\neq$ NT, SA $\neq$ Q, SA $\neq$ NSW, SA $\neq$ V, WA $\neq$ NT, NT $\neq$ Q, Q $\neq$ NSW, NSW $\neq$ V}

Constraint Graph:



CSP Example: Sudoku (3x3 for now)

Problem:

$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
$x_{3,1}$	$x_{3,2}$	$x_{3,3}$

Variables:

$X = \{x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, x_{2,2}, x_{2,3}, x_{3,1}, x_{3,2}, x_{3,3}\}$

Variable Domains:

$D_{x1,1} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$D_{x1,2} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$D_{x1,3} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$D_{x2,1} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$D_{x2,2} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$D_{x2,3} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$D_{x3,1} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$D_{x3,2} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$D_{x3,3} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

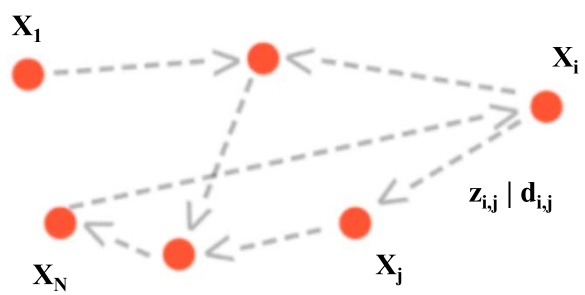
Constraints (Rules):

- **Each value {1, 2, 3, 4, 5, 6, 7, 8, 9} can appear EXACTLY once:**

CONSTRAINTS = C = $\{x_{1,1} \neq x_{1,2}, x_{1,1} \neq x_{1,3}, x_{1,1} \neq x_{2,1}, x_{1,1} \neq x_{2,2}, x_{1,1} \neq x_{2,3}, x_{1,2} \neq x_{1,3}, x_{1,2} \neq x_{2,1}, x_{1,2} \neq x_{2,2}, x_{1,2} \neq x_{2,3}, x_{1,2} \neq x_{3,1}, x_{1,2} \neq x_{3,2}, x_{1,3} \neq x_{2,1}, x_{1,3} \neq x_{2,2}, x_{1,3} \neq x_{2,3}, x_{1,3} \neq x_{3,1}, x_{1,3} \neq x_{3,2}, x_{1,3} \neq x_{3,3}, x_{2,1} \neq x_{2,2}, x_{2,1} \neq x_{2,3}, x_{2,1} \neq x_{3,1}, x_{2,1} \neq x_{3,2}, x_{2,1} \neq x_{3,3}, x_{2,2} \neq x_{2,3}, x_{2,2} \neq x_{3,1}, x_{2,2} \neq x_{3,2}, x_{2,2} \neq x_{3,3}, x_{2,3} \neq x_{3,1}, x_{2,3} \neq x_{3,2}, x_{2,3} \neq x_{3,3}, x_{3,1} \neq x_{3,2}, x_{3,1} \neq x_{3,3}, x_{3,2} \neq x_{3,3}\}$

CSP Example: Traveling Salesman

Problem:



There are:

- N cities (vertices)
- N(N-1) links (edges)
- Each link has some positive cost d
- Total path (tour) cost is COST

Variables:

$Z = \{z_{1,2}, z_{1,3}, \ldots, z_{N-1,N}\}$
$D = \{d_{1,2}, d_{1,3}, \ldots, d_{N-1,N}\}$

Variable Domains:

$D_{zi,j} = \{traveled, notTraveled\}$

or better:

$D_{zi,j} = \{1, 0\}$

$D_{di,j} = R_+$

Constraints (Rules):

- Exit each city EXACTLY once:

$$\sum_{j=1}^{N} z_{i,j} = 1$$

- Enter each city EXACTLY once:

$$\sum_{i=1}^{N} z_{i,j} = 1$$

- Cost of tour is at most C:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} z_{i,j} d_{i,j} \leq COST$$

CSP: Variable Types

- Domains can be:
  - finite, for example: $\{1, 2, 3, 5, 8, 20\}$ (simpler)
  - infinite, for example: a set of all integers
- Variables can be:
  - discrete, for example: $X = \{X_1, ..., X_n\}$ (simpler)
  - continuous, for example: R_+
- Constraints can be:
  - unary (involve single variable), for example: $X_1 = 5$
  - binary (involve two variables), for example: $X_1 = X_2$
  - higher order (involve > 2 variables), for example: $X_1 = X_2 * X_3$
- Soft constraints (preferences: green over blue) possible

CSP as a Search Problem

CSP is a variant of a search problem you already know. The problem can be restated / updated with:

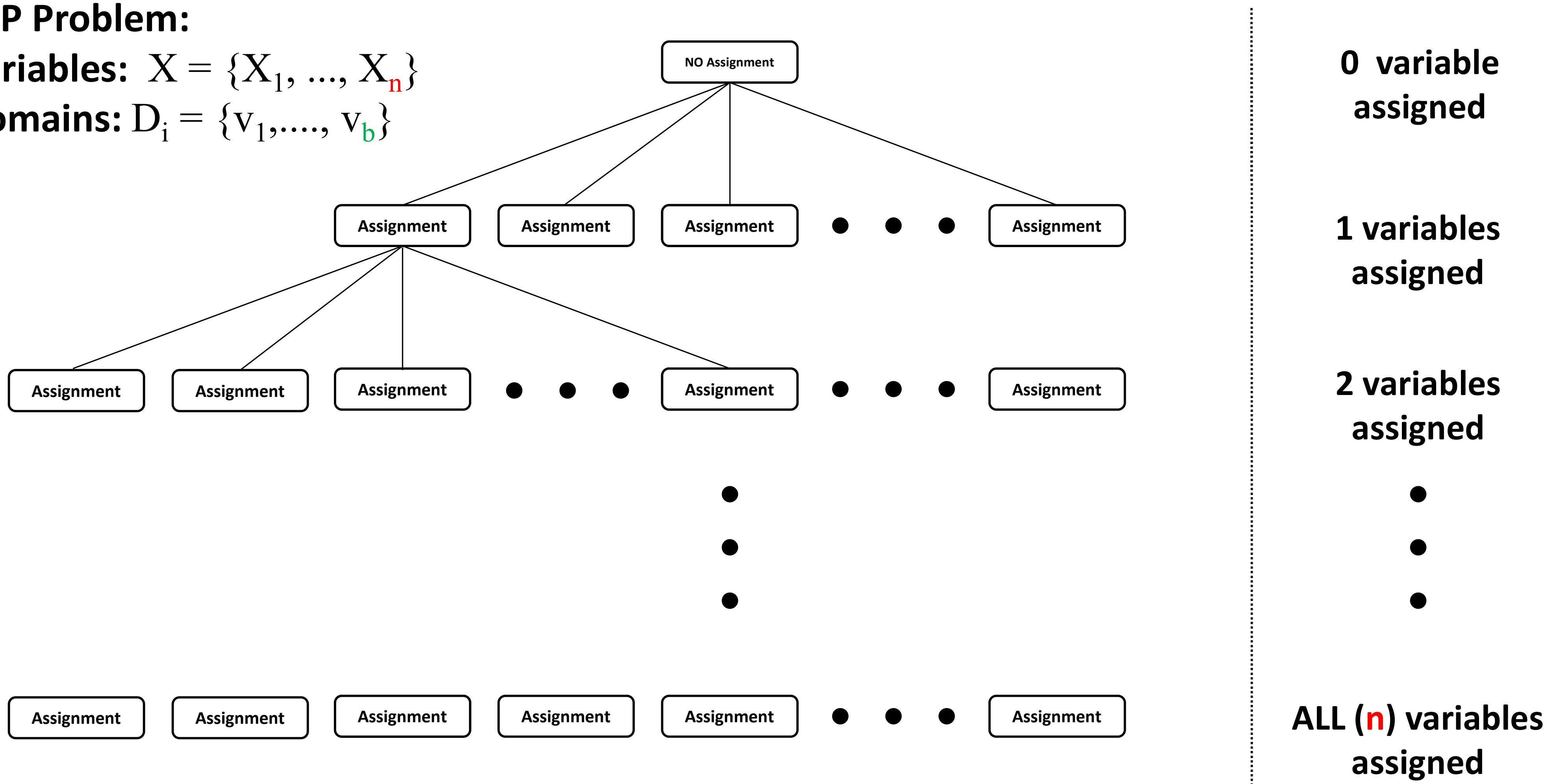
- **Initial state: the empty assignment { }, in which all variables are unassigned.**
- **Successor function: a value can be assigned to any unassigned variable, provided that it does not conflict with previously assigned variables.**
- **Goal test: the current assignment is complete.**
- **Path cost: a constant cost (e.g., 1) for every step.**

CSP Search Tree: Idea

CSP Problem:

Variables: $X = \{X_1, \ldots, X_n\}$

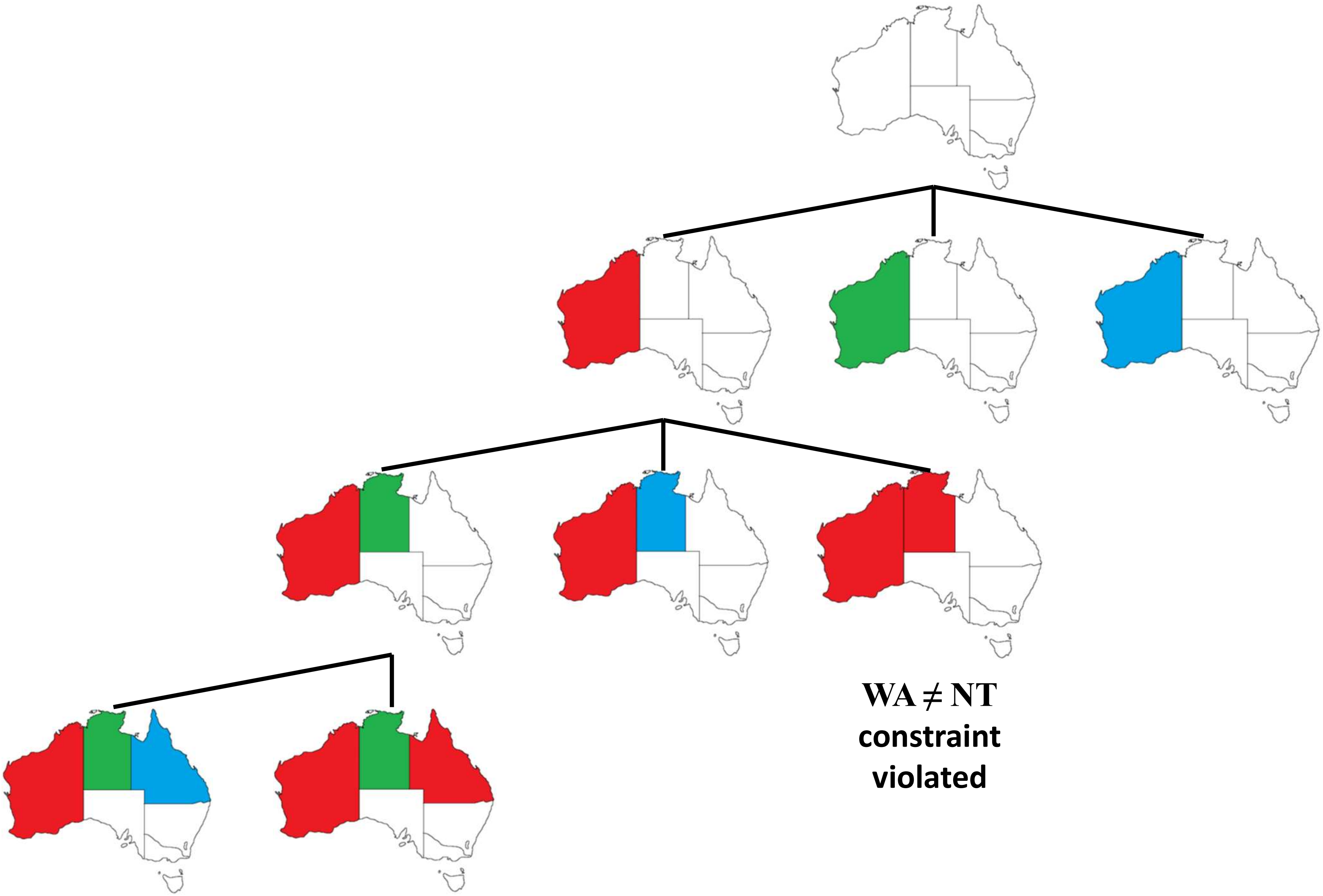
Domains: $D_i = \{v_1, \ldots, v_b\}$



Tree leaves are COMPLETE assignments

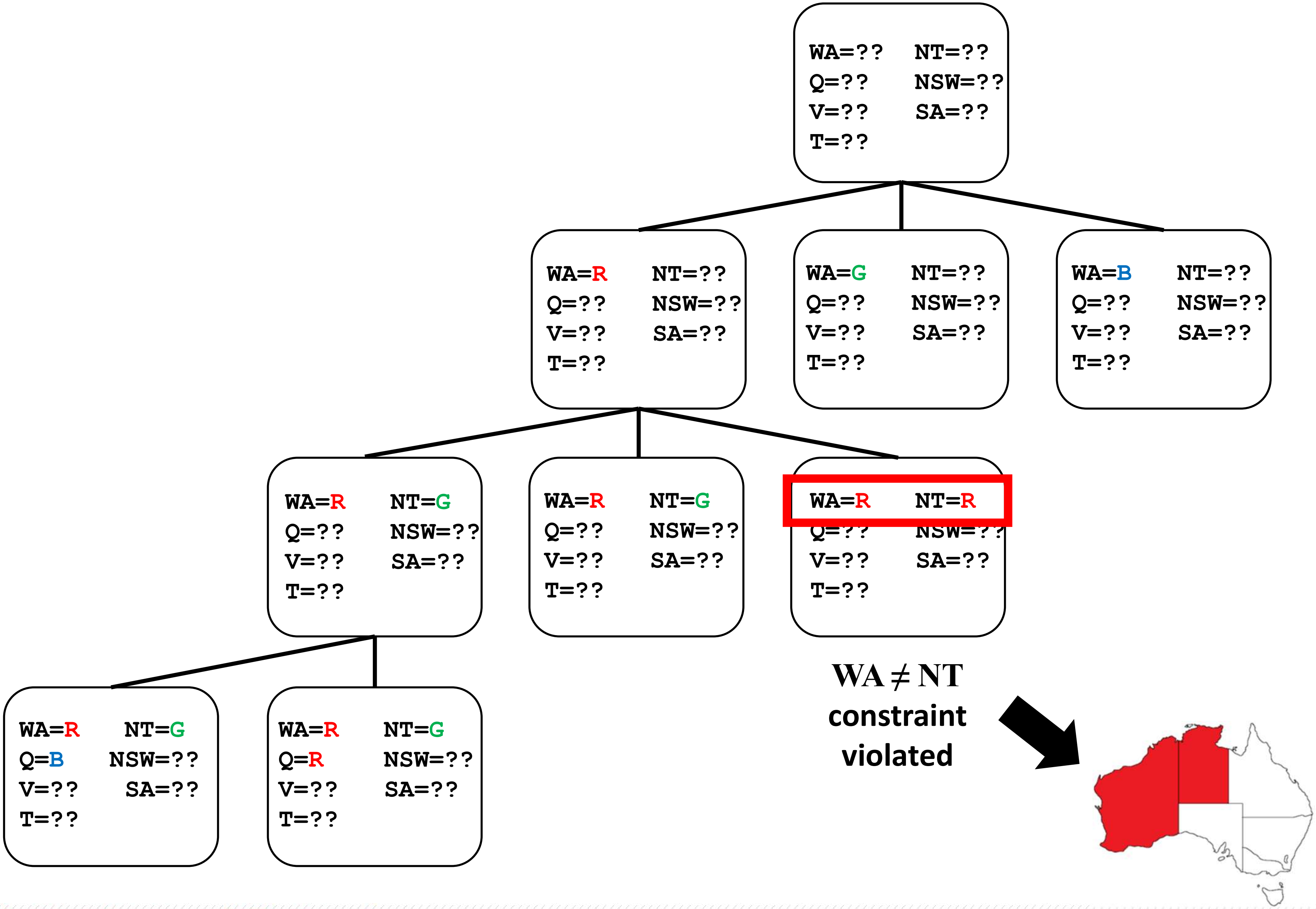
The sequence of variable assignments does NOT matter*
*(when you disregard performance)

CSP as a Tree Search Problem



WA ≠ NT constraint violated

CSP as a Tree Search Problem

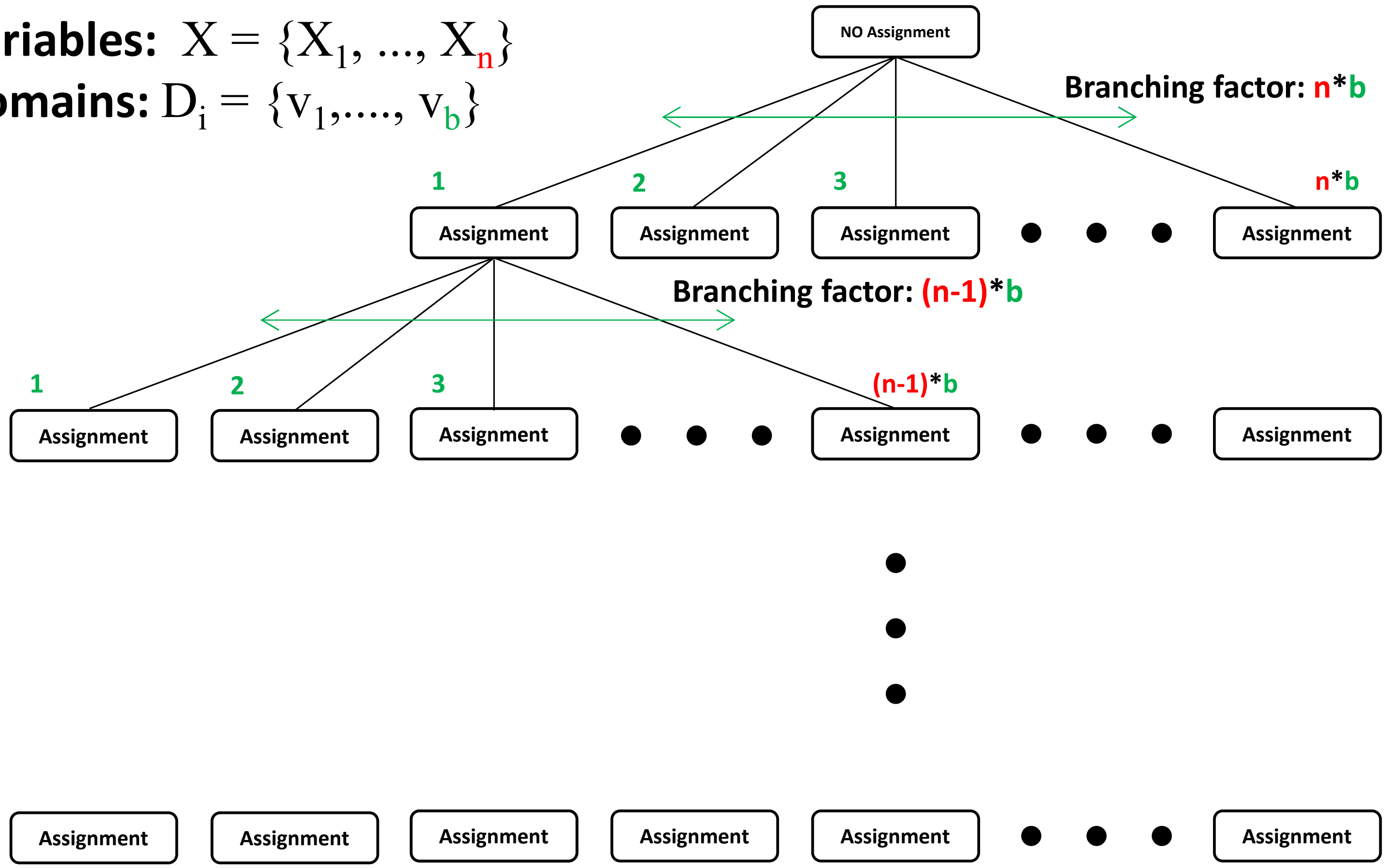


CSP Search Tree: Size

CSP Problem:

Variables: $X = \{X_1, \ldots, X_n\}$

Domains: $D_i = \{v_1, \ldots, v_b\}$



$N_0 = 0$

$N_1 = n*b$

$N_2 = n*b*(n-1)*b =$
$= n*(n-1)*b^2$

$N_n = n! * b^n$

Total number of leafnodes / states: $n! * b^n$

(ignores COMMUTATIVITY of CSP assignments:

assigning $X_1 = m$ and then $X_2 = n$ SAME as assigning $X_2 = n$ and then $X_1 = m$)

<u>In reality: there is only b^n complete assignments</u>

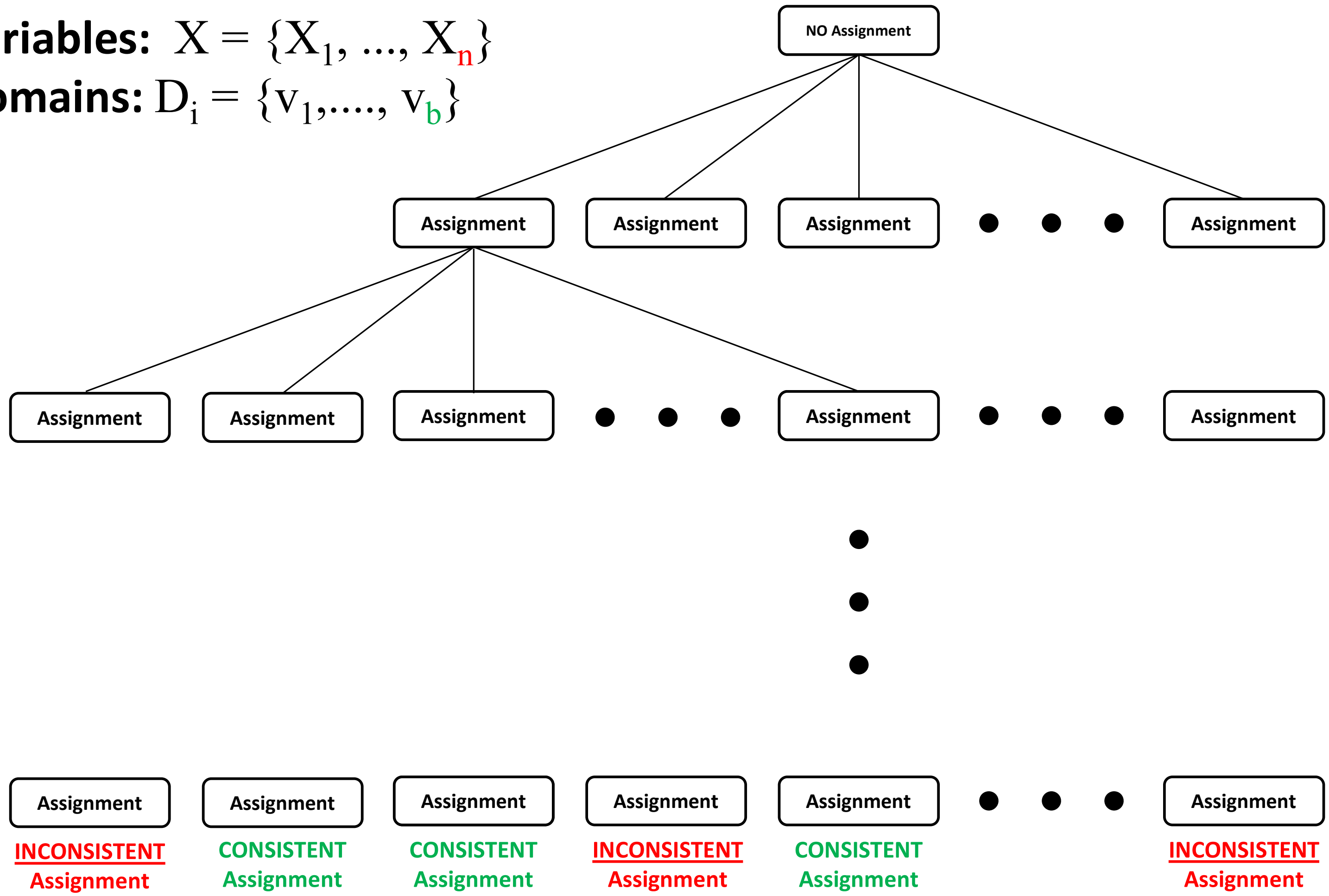
Can We Do Better?

CSP Search Tree: Solutions

CSP Problem:

Variables: $X = \{X_1, \ldots, X_n\}$

Domains: $D_i = \{v_1, \ldots, v_b\}$

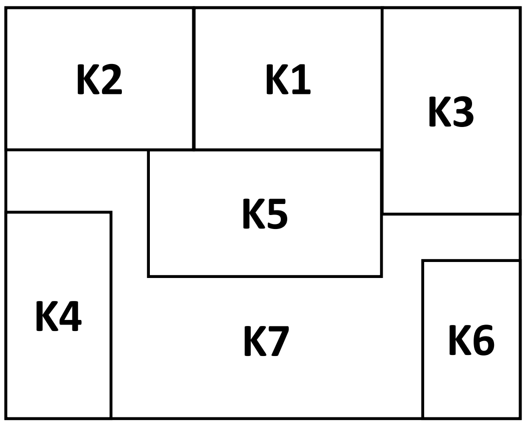


Some nodes / states will be CONSISTENT, while others will be INCONSISTENT.

Depth first search could possibly visit them all → WASTEFUL.

CSP Example: Map Coloring

Problem:



Color this map in a way that no two neighbors have same color

Variables:

$X = \{K1, K2, K3, K4, K5, K6, K7\}$

Variable Domains:

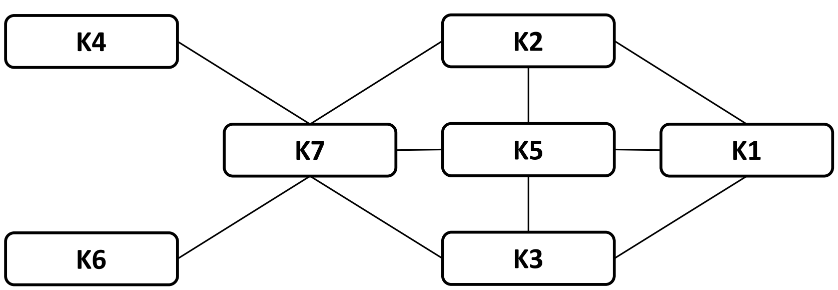
D_{K1} = {RED, BLUE, GREEN}
D_{K2} = {RED, BLUE, GREEN}
D_{K3} = {RED, BLUE, GREEN}
D_{K4} = {RED, BLUE, GREEN}
D_{K5} = {RED, BLUE, GREEN}
D_{K6} = {RED, BLUE, GREEN}
D_{K7} = {RED, BLUE, GREEN}

Constraints (Rules):

- **Neighboring regions have to have DISTINCT colors:**

$CONSTRAINTS = C = \{K1 \neq K2, K1 \neq K3, K1 \neq K5, K2 \neq K5, K2 \neq K7, K3 \neq K5, K3 \neq K7, K4 \neq K7, K5 \neq K7, K6 \neq K7\}$

Constraint Graph:



CSP Backtracking: Pseudocode

```
function BACKTRACKING-SEARCH(csp) returns a solution or failure
  return BACKTRACK(csp, { })

function BACKTRACK(csp, assignment) returns a solution or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp, assignment)
  for each value in ORDER-DOMAIN-VALUES(csp, var, assignment) do
      if value is consistent with assignment then
         add {var = value} to assignment
         inferences ← INFERENCE(csp, var, assignment)
         if inferences ≠ failure then
            add inferences to csp
            result ← BACKTRACK(csp, assignment)
            if result ≠ failure then return result
            remove inferences from csp
         remove {var = value} from assignment
  return failure
```

CSP Backtracking: Pseudocode

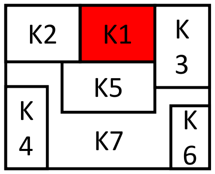
```
function BACKTRACKING-SEARCH(csp) returns a solution or failure
  return BACKTRACK(csp, { })

function BACKTRACK(csp, assignment) returns a solution or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp, assignment)
  for each value in ORDER-DOMAIN-VALUES(csp, var, assignment) do
      if value is consistent with assignment then
        add {var = value} to assignment
        inferences ← INFERENCE(csp, var, assignment)
        if inferences ≠ failure then
          add inferences to csp
          result ← BACKTRACK(csp, assignment)
          if result ≠ failure then return result
          remove inferences from csp
        remove {var = value} from assignment
  return failure
```

RECURSION

Assignment:
K1: RED
K2: ???
K3: ???
K4: ???
K5: ???
K6: ???
K7: ???

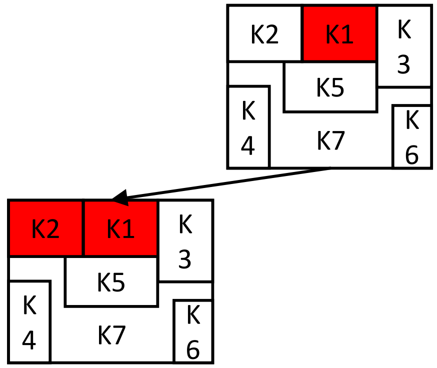
Initial (NO assignment) state not shown



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: RED
K3: ???
K4: ???
K5: ???
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:

K1: RED

K2: RED

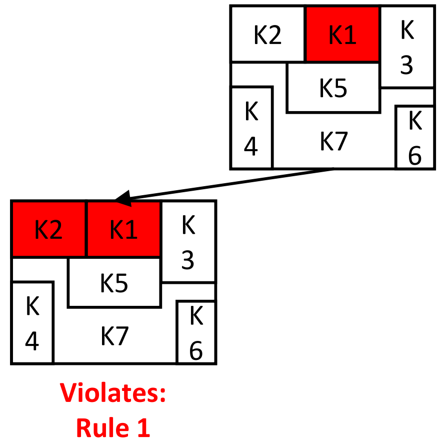
K3: ???

K4: ???

K5: ???

K6: ???

K7: ???



Constraints:

Rule 1: K1 ≠ K2

Rule 2: K1 ≠ K3

Rule 3: K1 ≠ K5

Rule 4: K2 ≠ K5

Rule 5: K2 ≠ K7

Rule 6: K3 ≠ K5

Rule 7: K3 ≠ K7

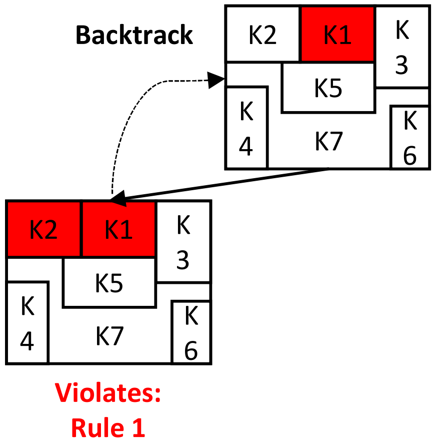
Rule 8: K4 ≠ K7

Rule 9: K5 ≠ K7

Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

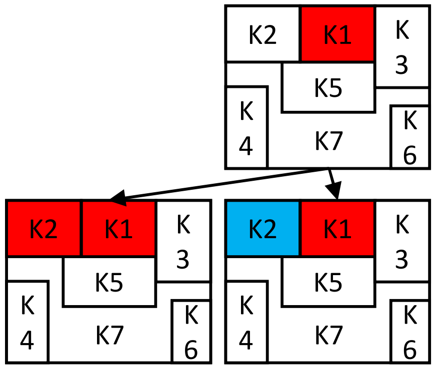
Assignment:
K1: RED
K2: ???
K3: ???
K4: ???
K5: ???
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

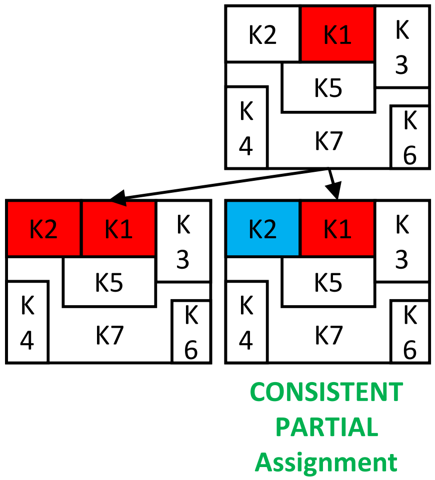
Assignment:
K1: RED
K2: BLUE
K3: ???
K4: ???
K5: ???
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order:** RED, BLUE, GREEN

Assignment:
K1: RED
K2: BLUE
K3: ???
K4: ???
K5: ???
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:

K1: RED

K2: BLUE

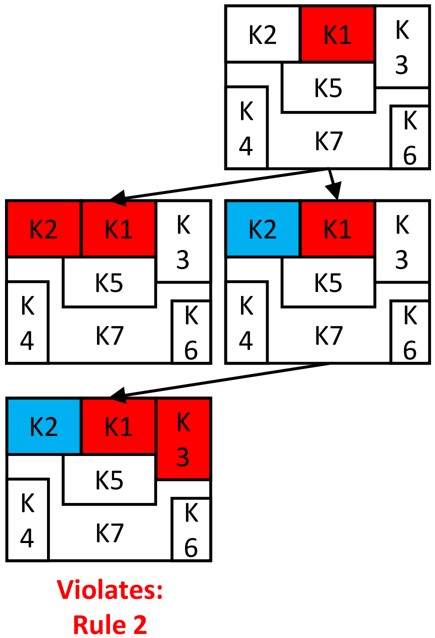
K3: RED

K4: ???

K5: ???

K6: ???

K7: ???



Constraints:

Rule 1: K1 ≠ K2

Rule 2: K1 ≠ K3

Rule 3: K1 ≠ K5

Rule 4: K2 ≠ K5

Rule 5: K2 ≠ K7

Rule 6: K3 ≠ K5

Rule 7: K3 ≠ K7

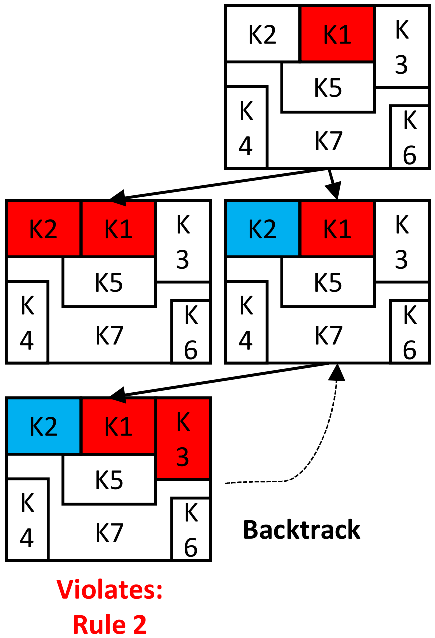
Rule 8: K4 ≠ K7

Rule 9: K5 ≠ K7

Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

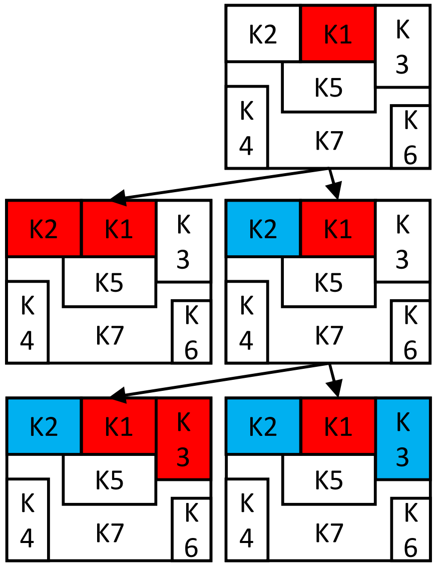
Assignment:
K1: RED
K2: BLUE
K3: ???
K4: ???
K5: ???
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

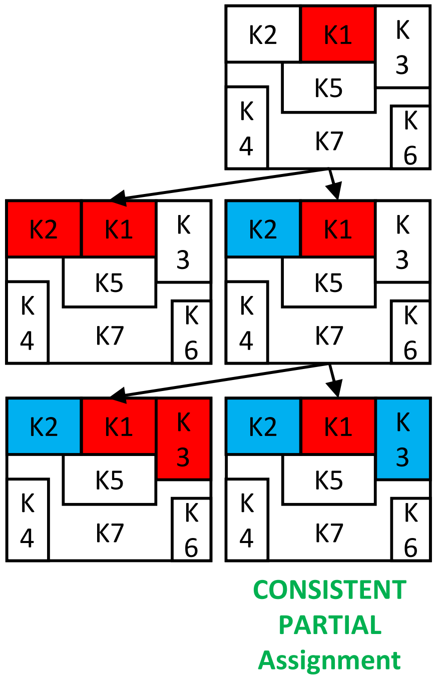
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: ???
K5: ???
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

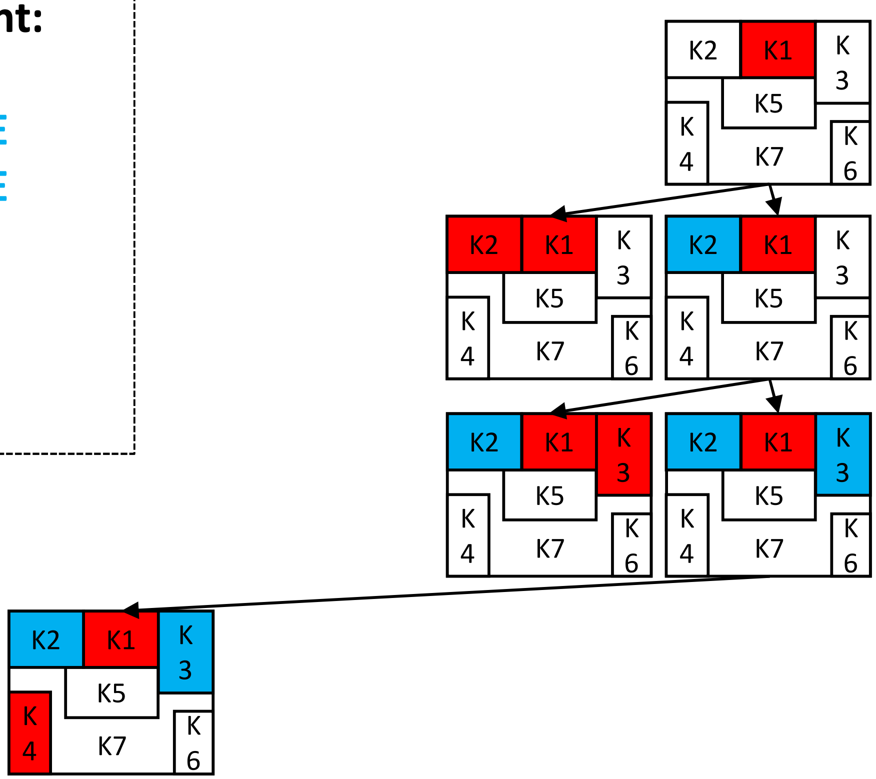
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: ???
K5: ???
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: ???
K6: ???
K7: ???

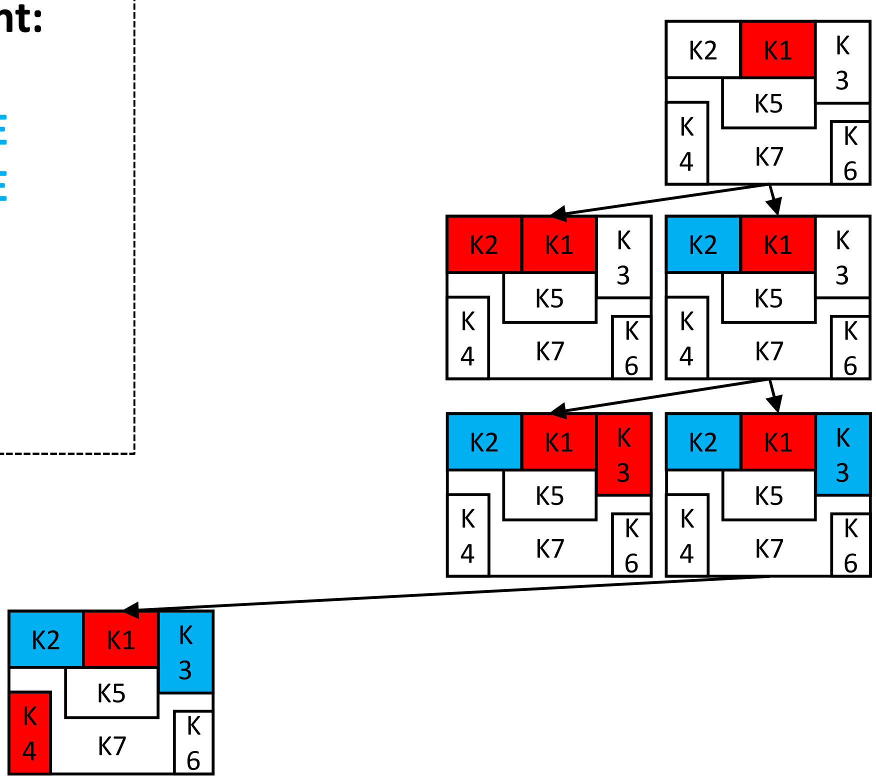


CONSISTENT PARTIAL Assignment

Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

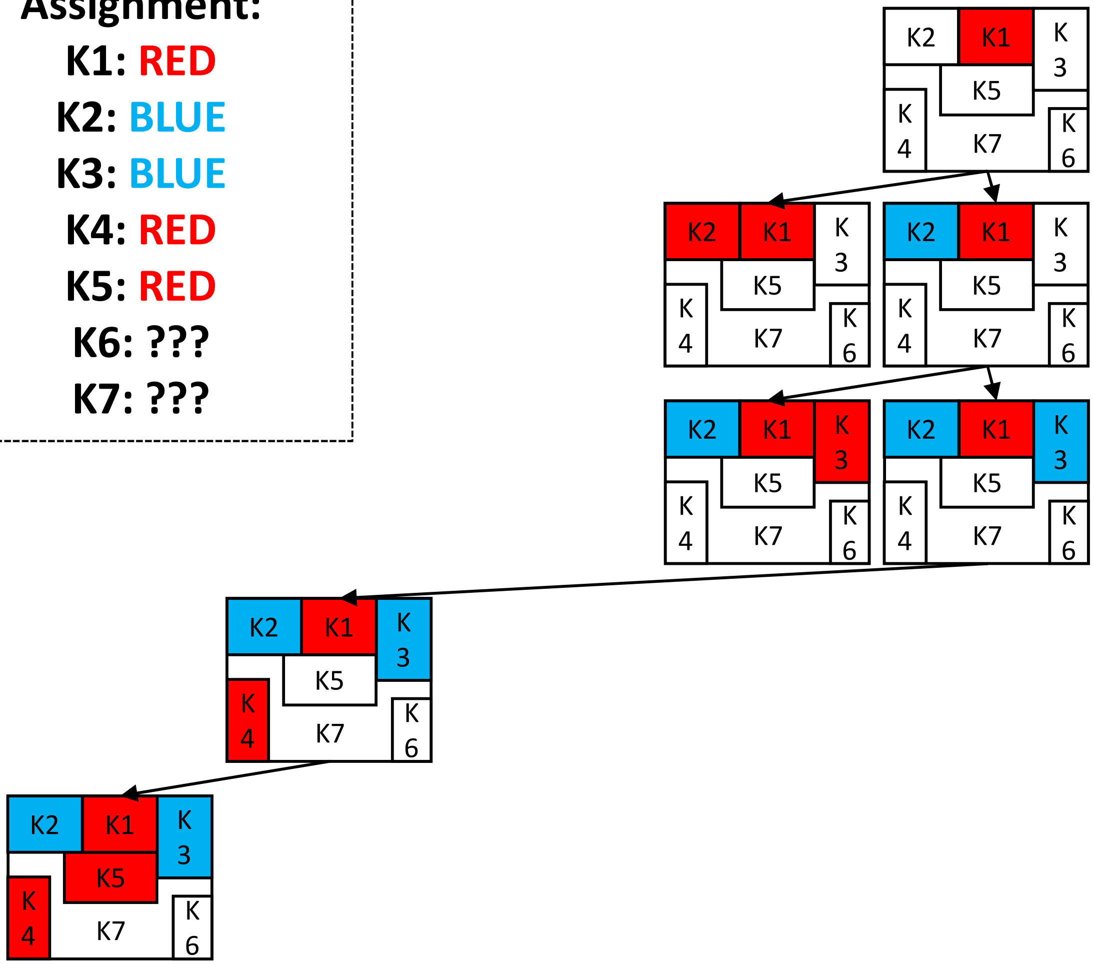
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: ???
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

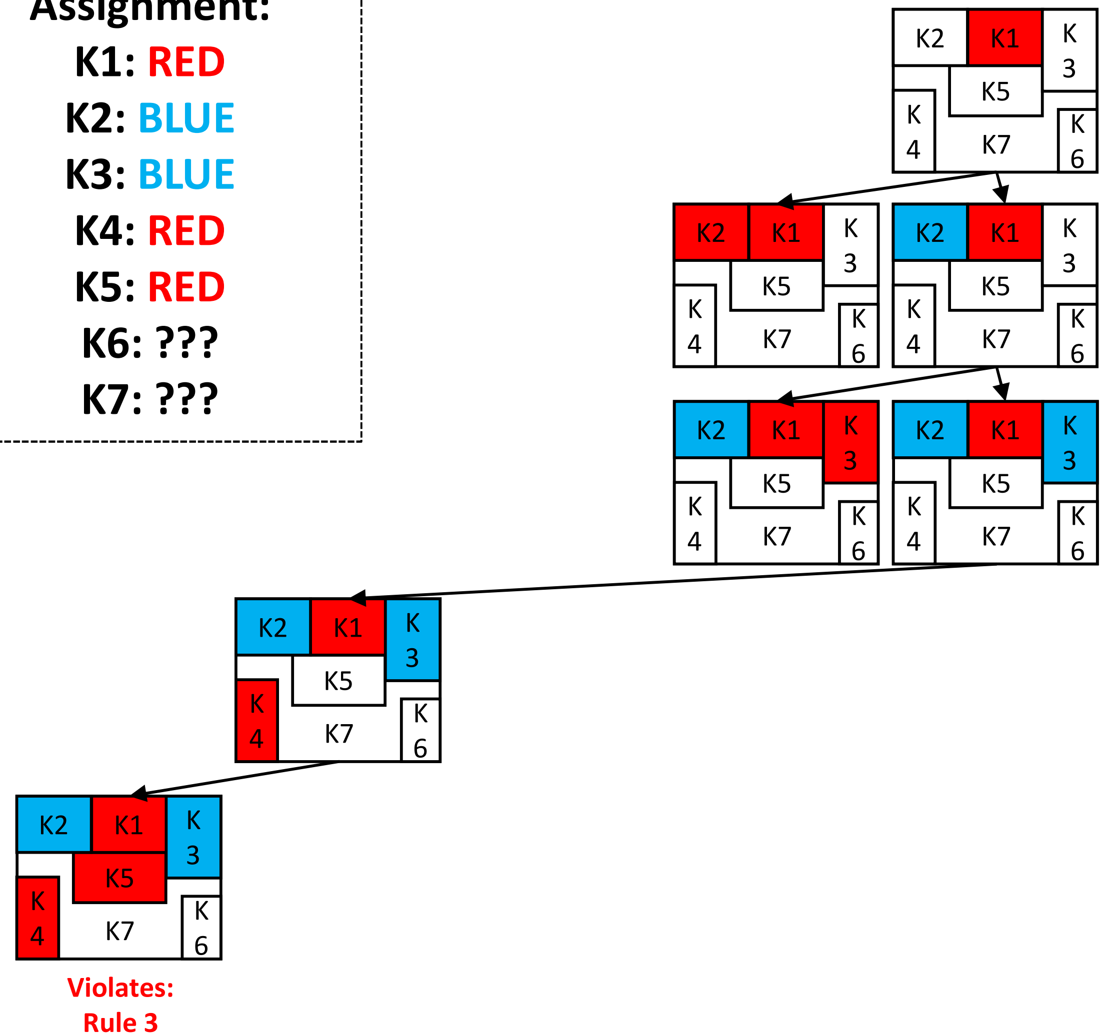
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: RED
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

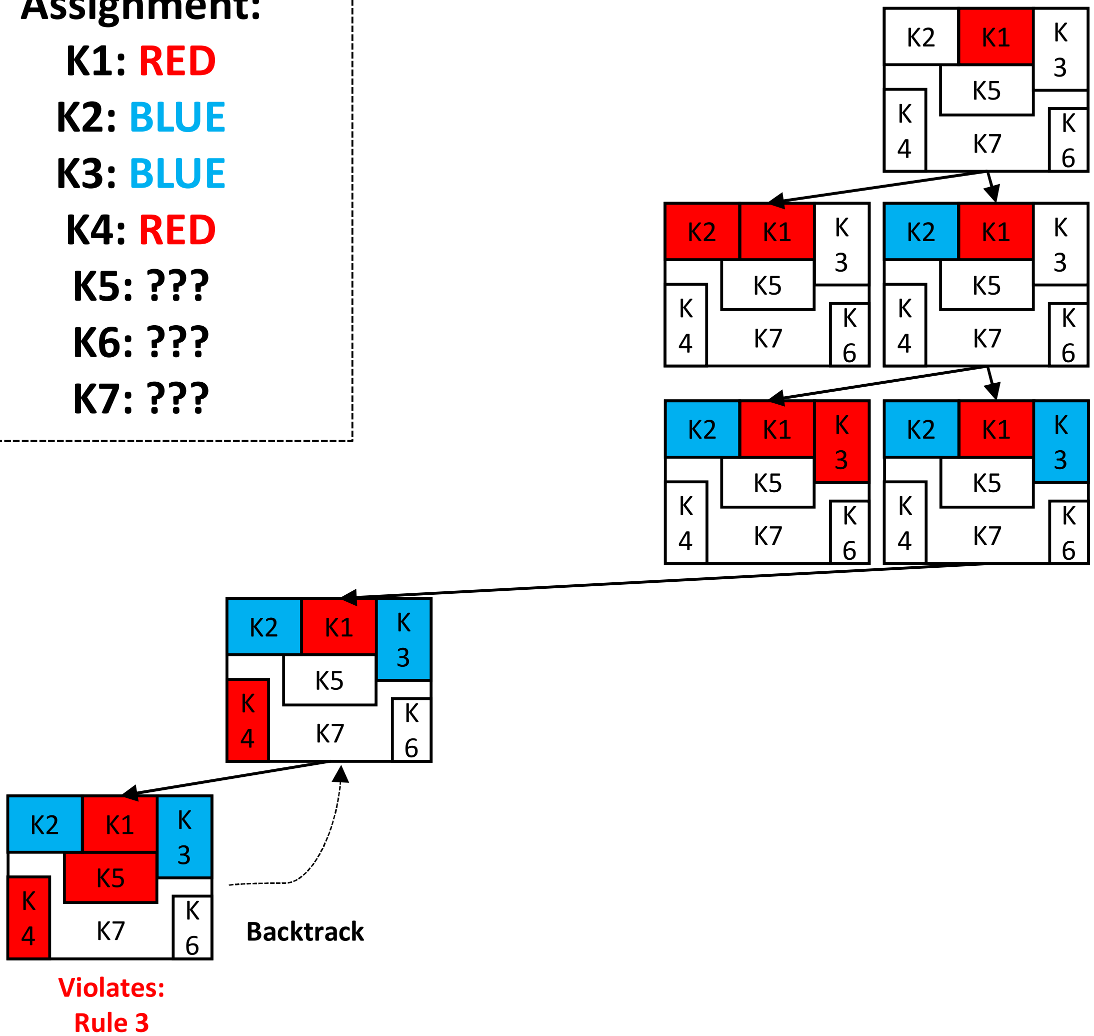
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: RED
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

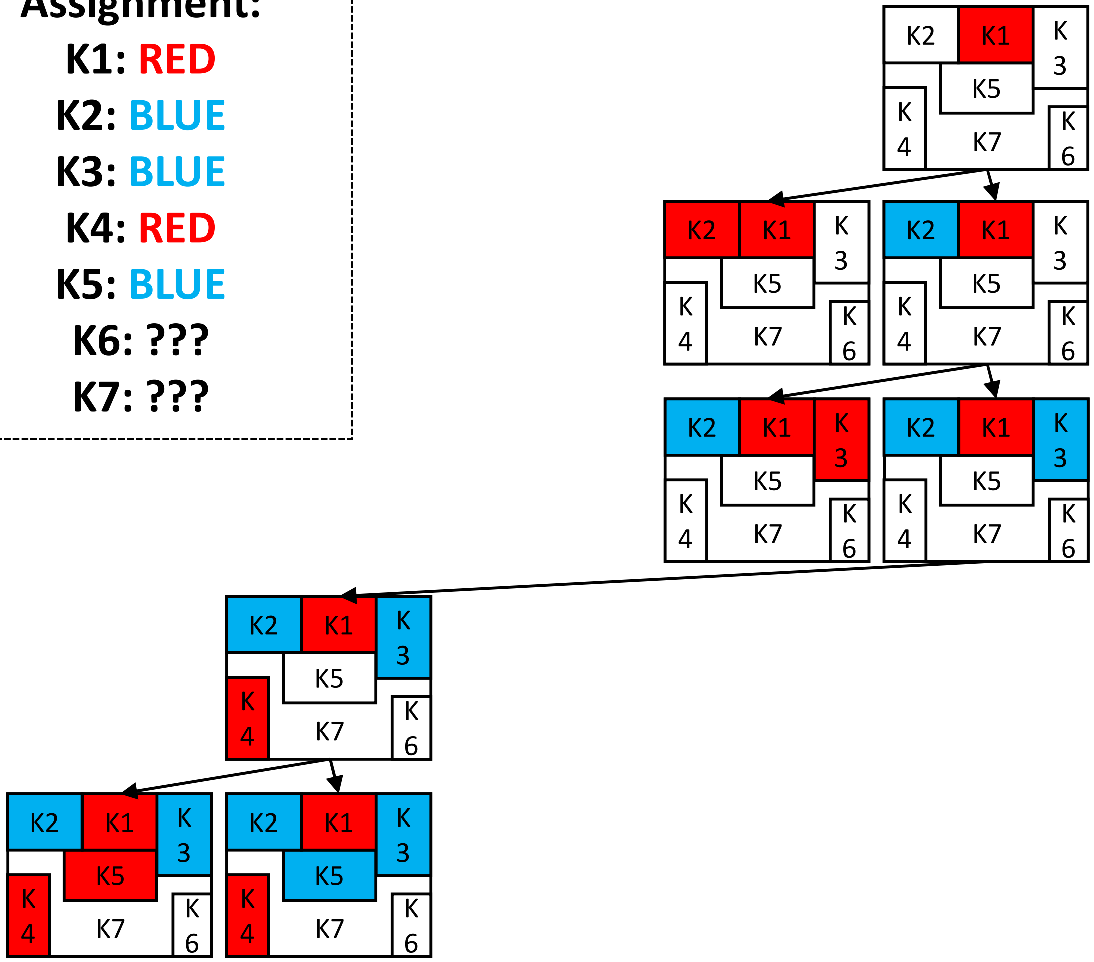
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: ???
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

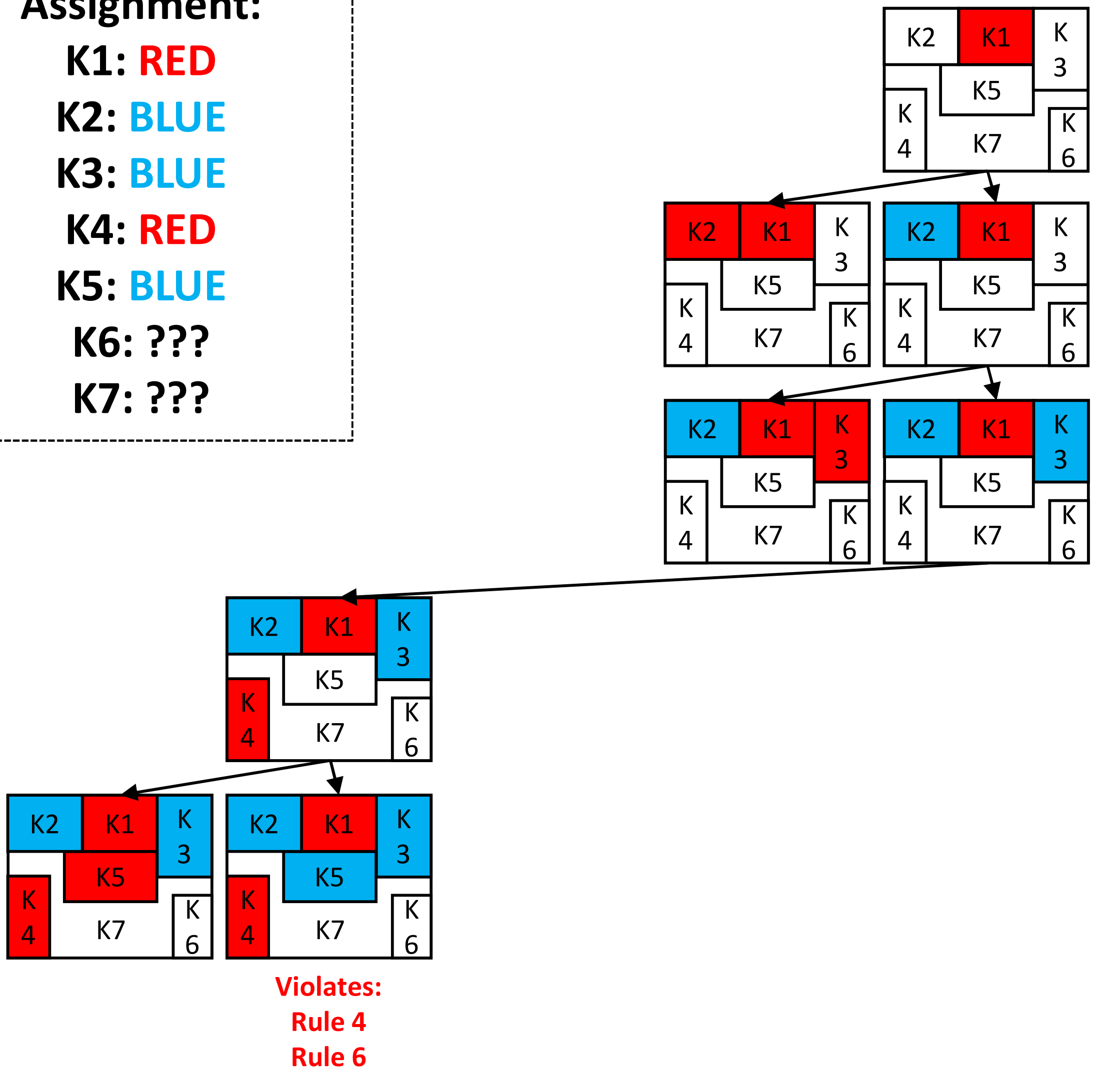
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: BLUE
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

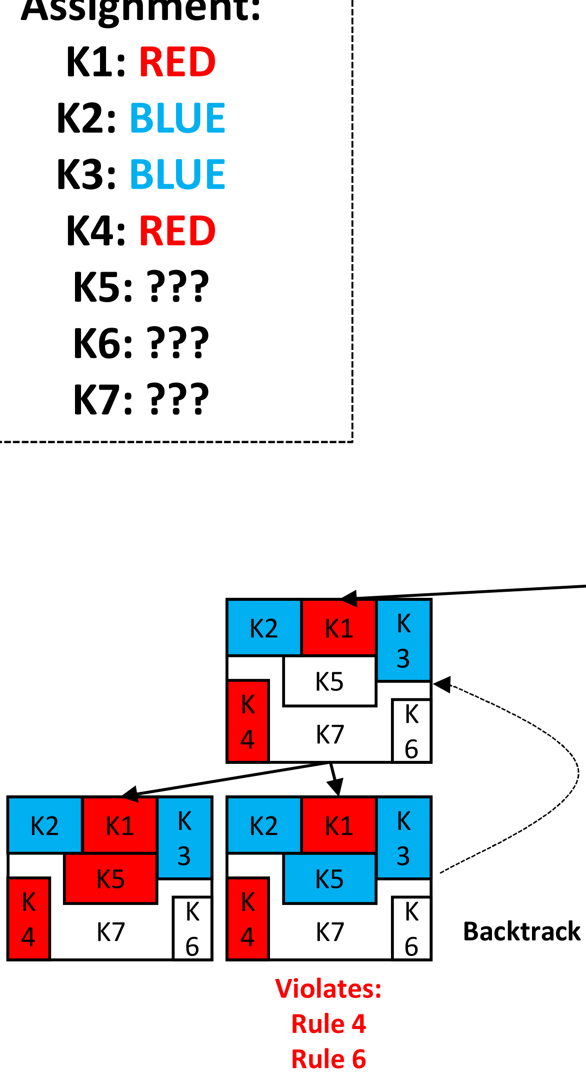
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: BLUE
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order:** RED, BLUE, GREEN

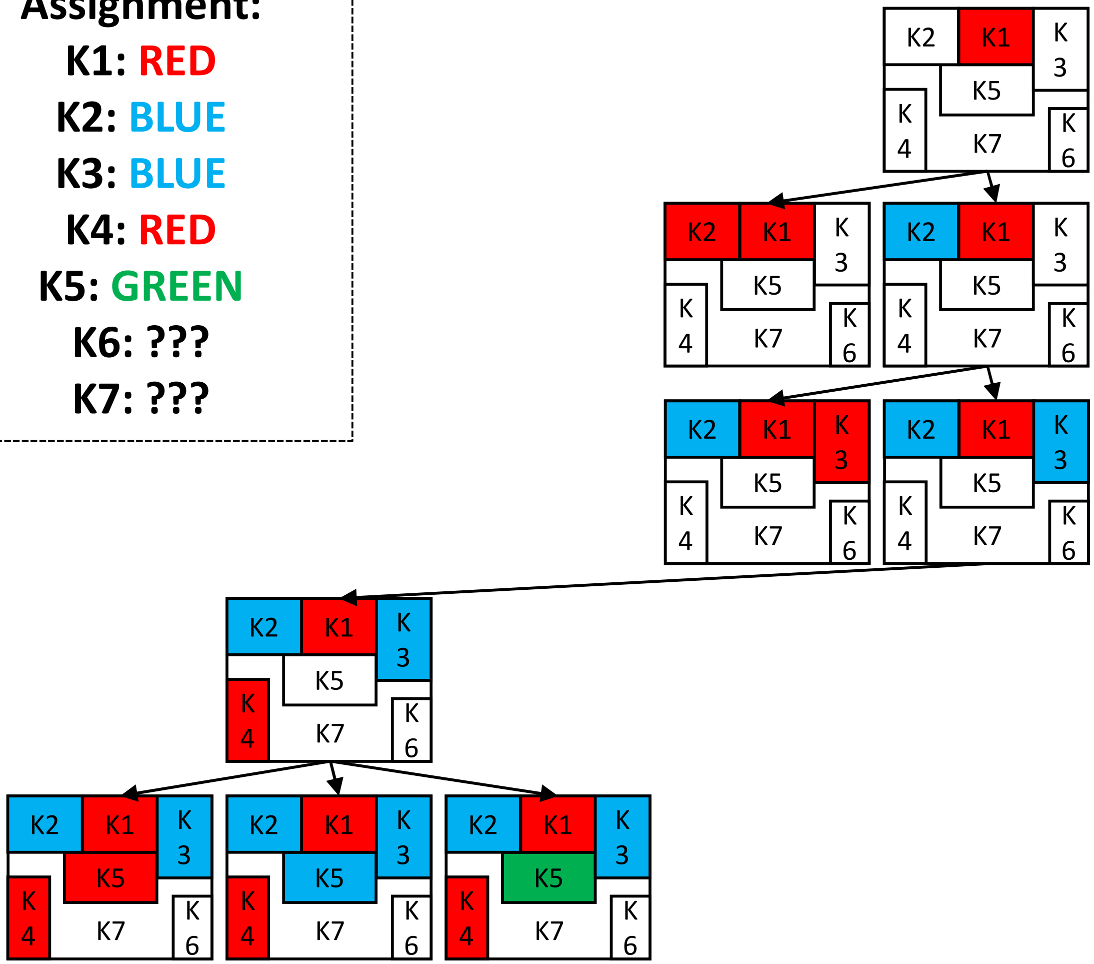
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: ???
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order:** RED, BLUE, GREEN

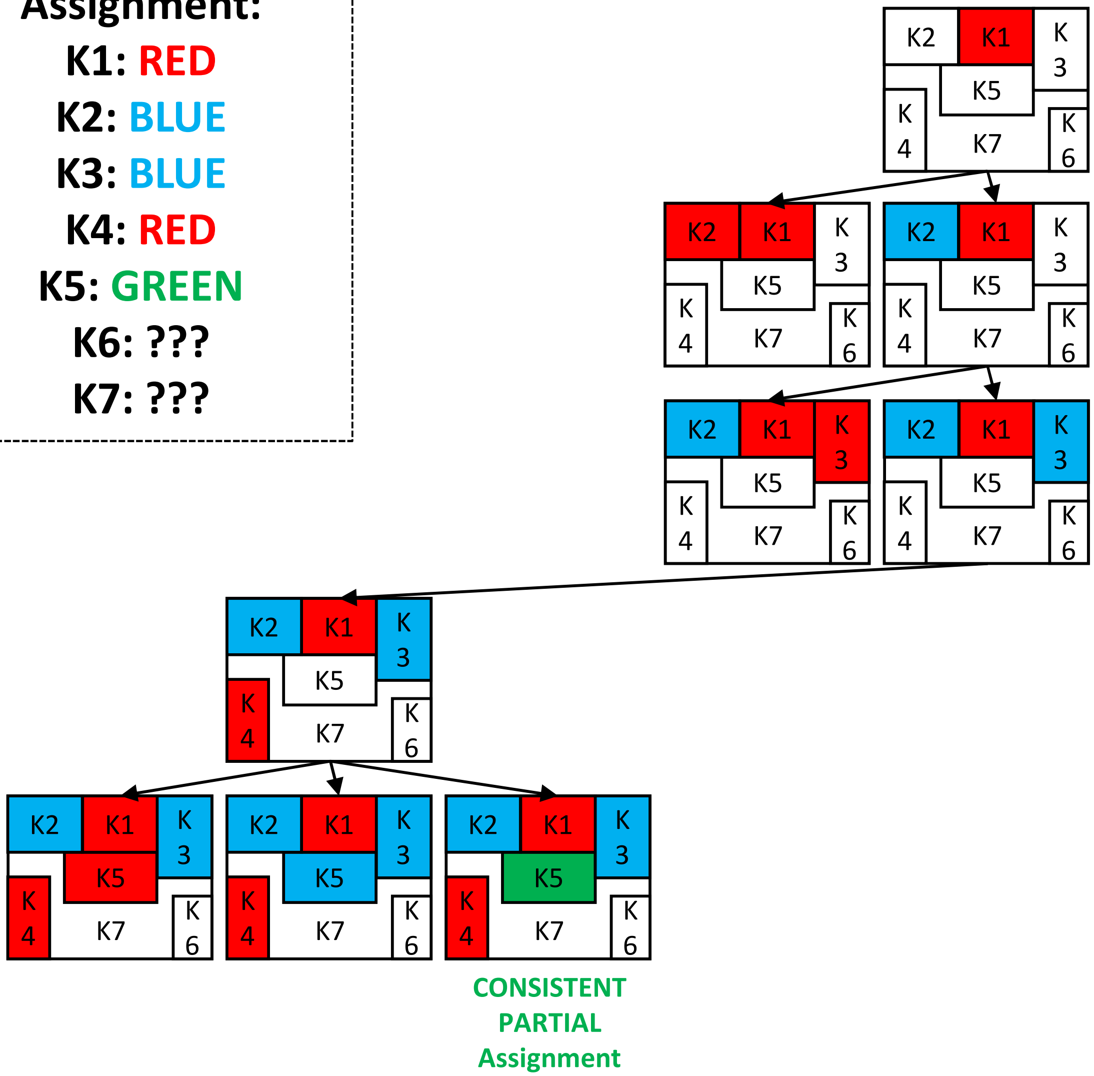
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

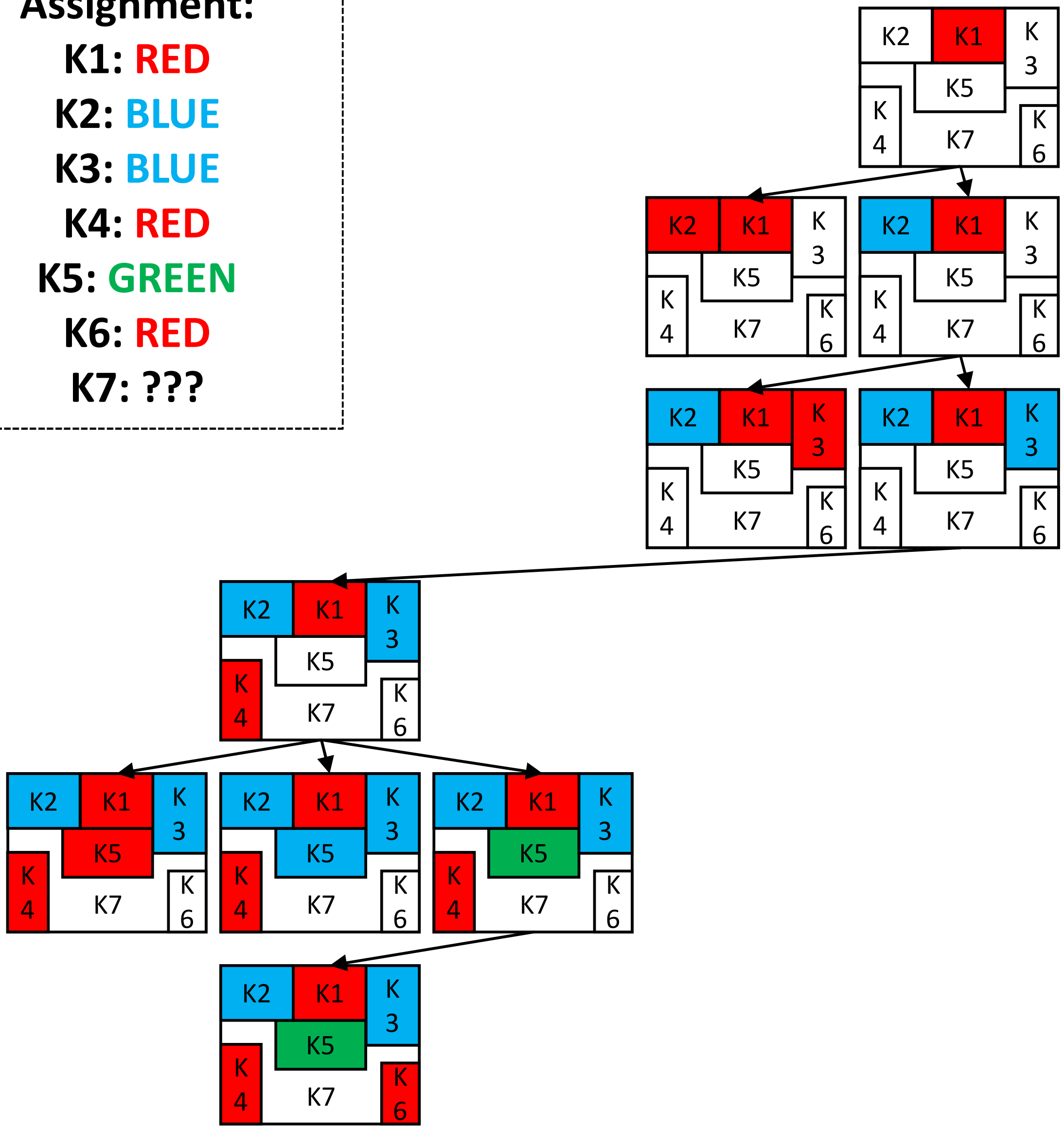
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

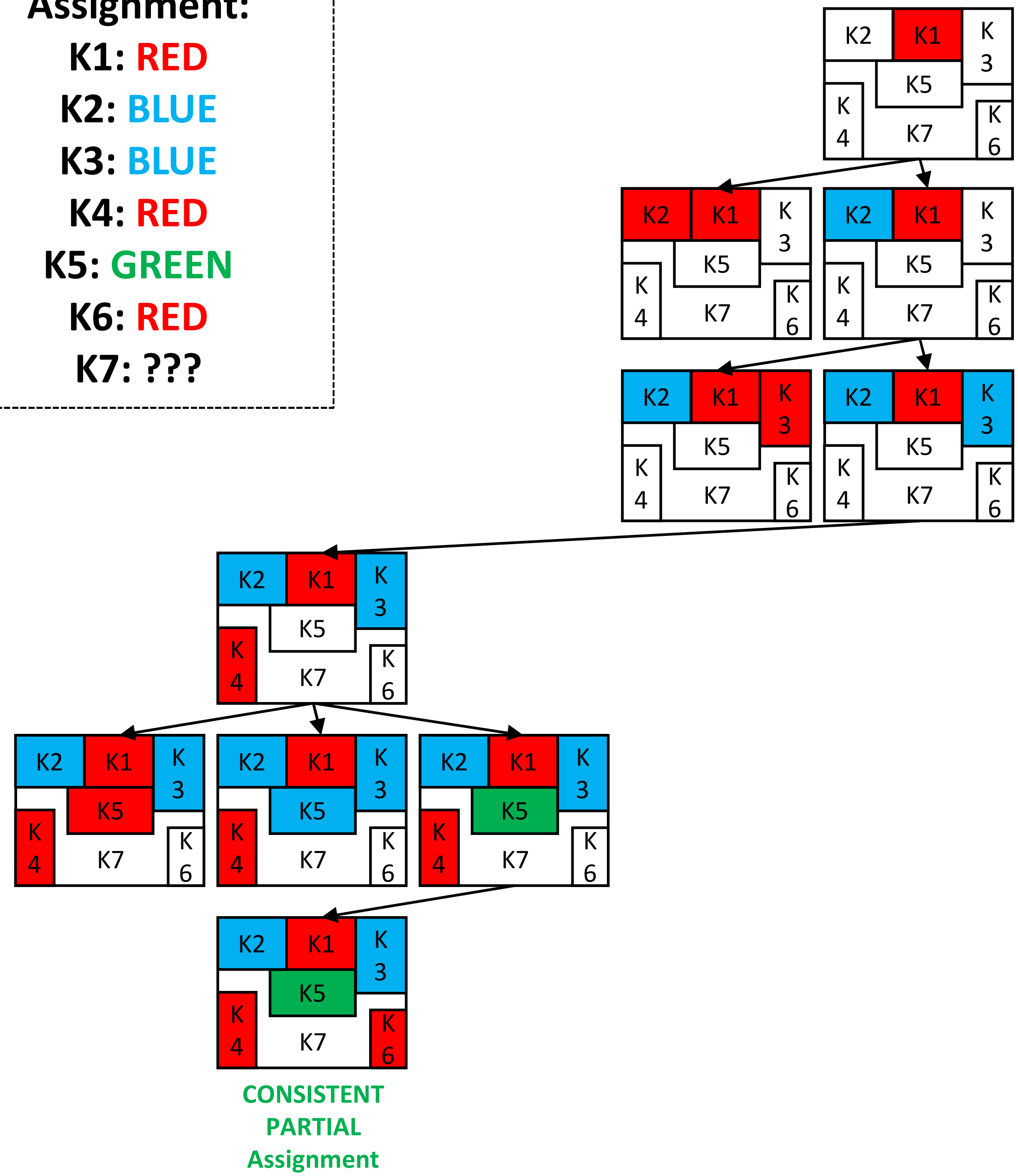
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: RED
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

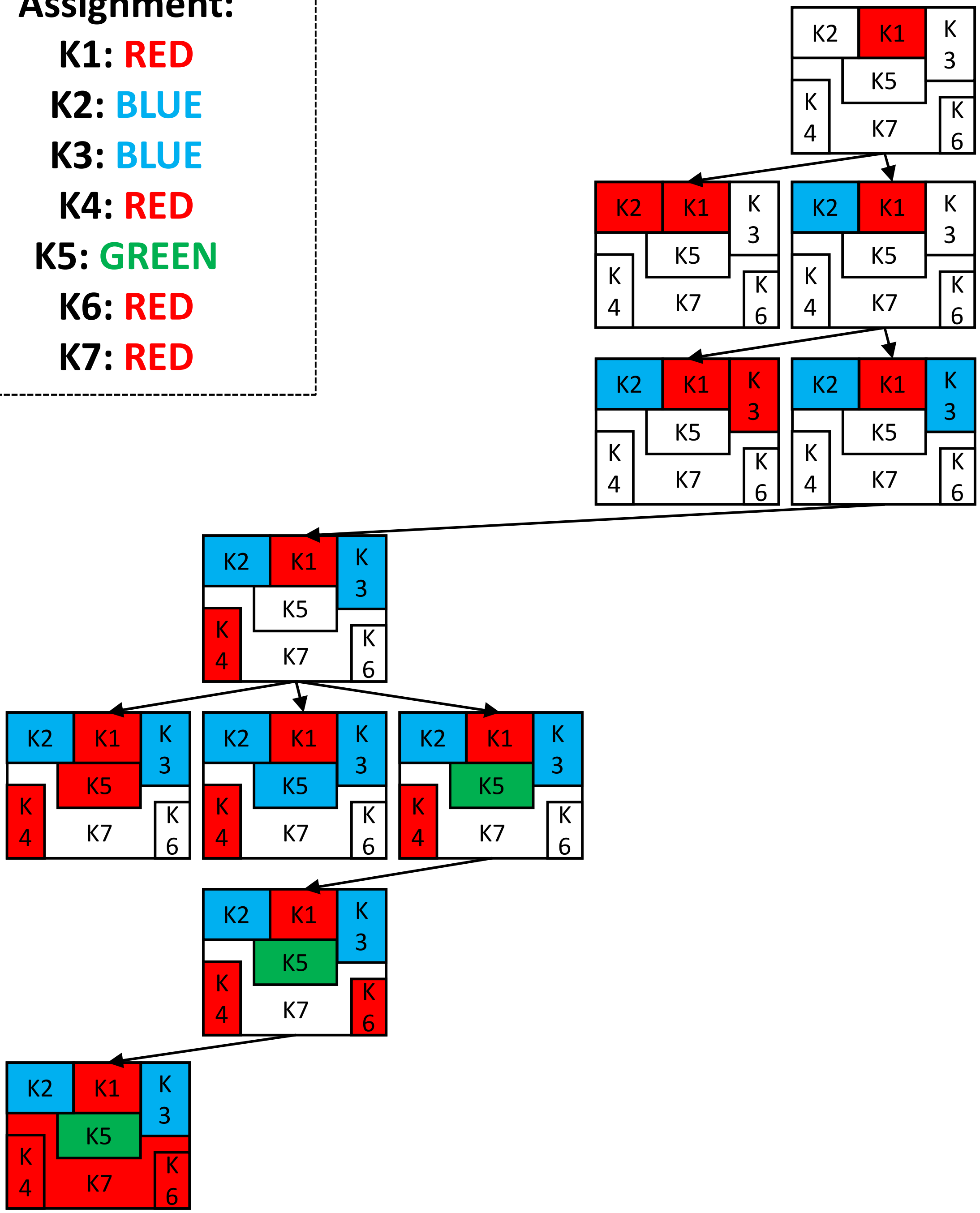
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: RED
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

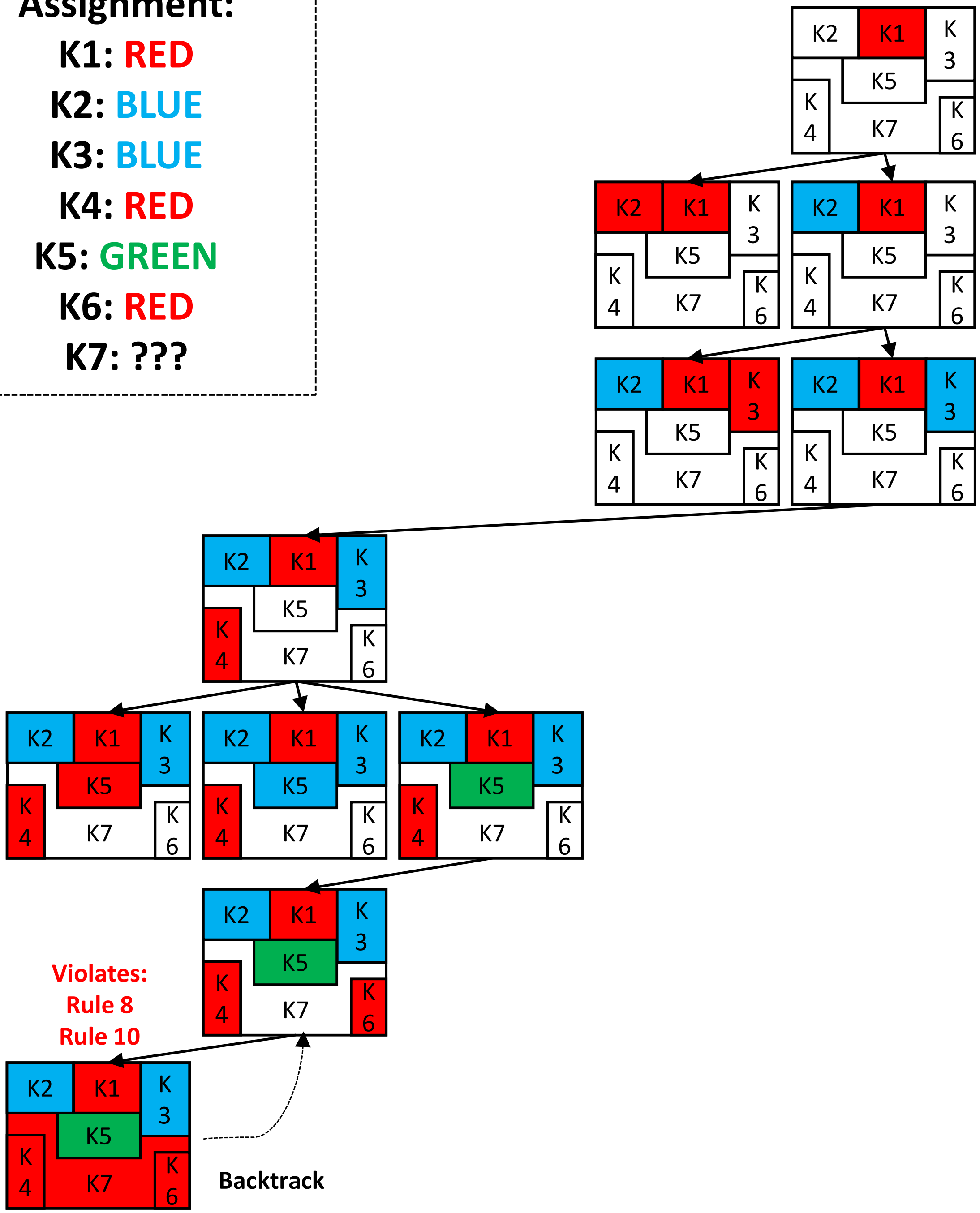
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: RED
K7: RED



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

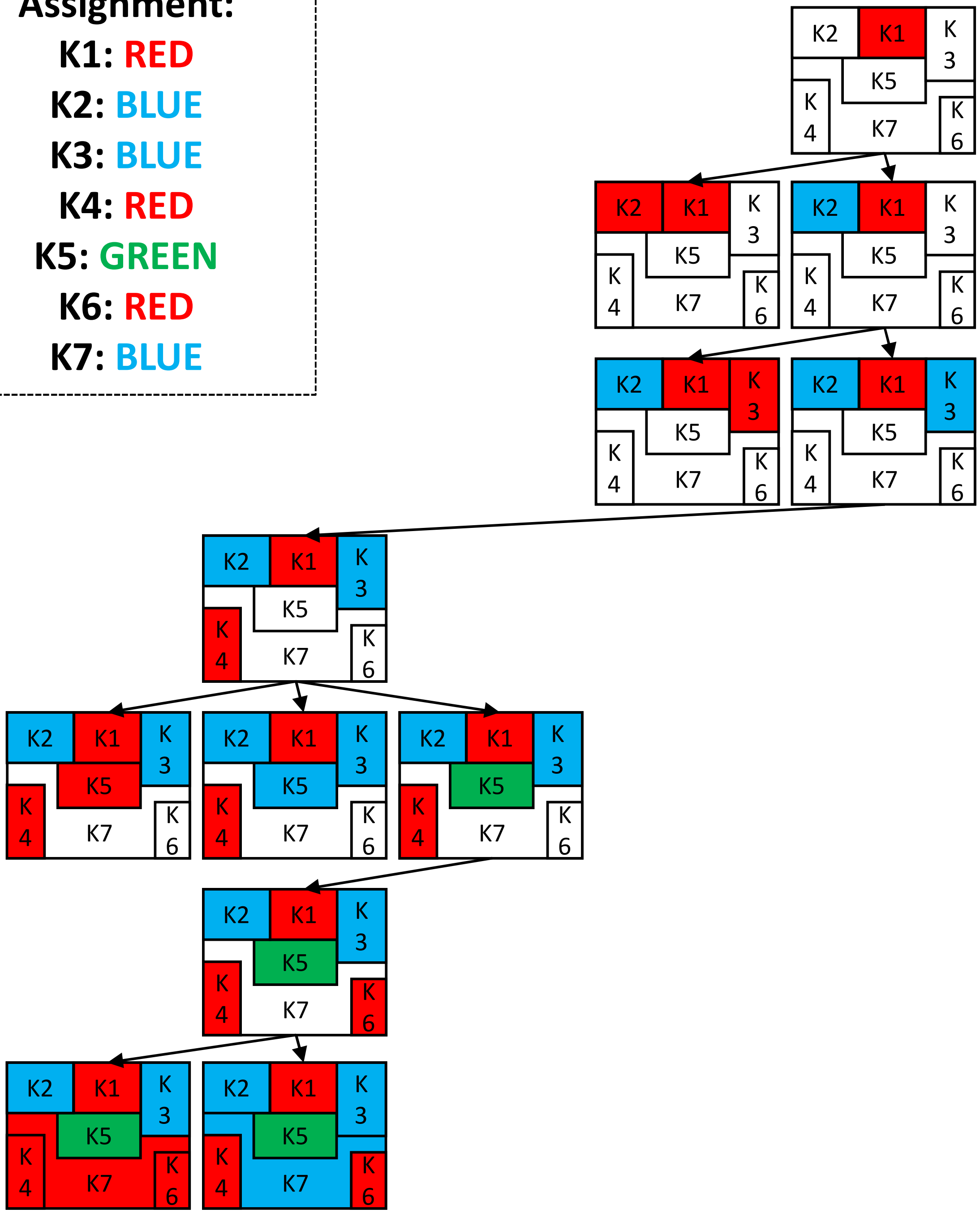
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: RED
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

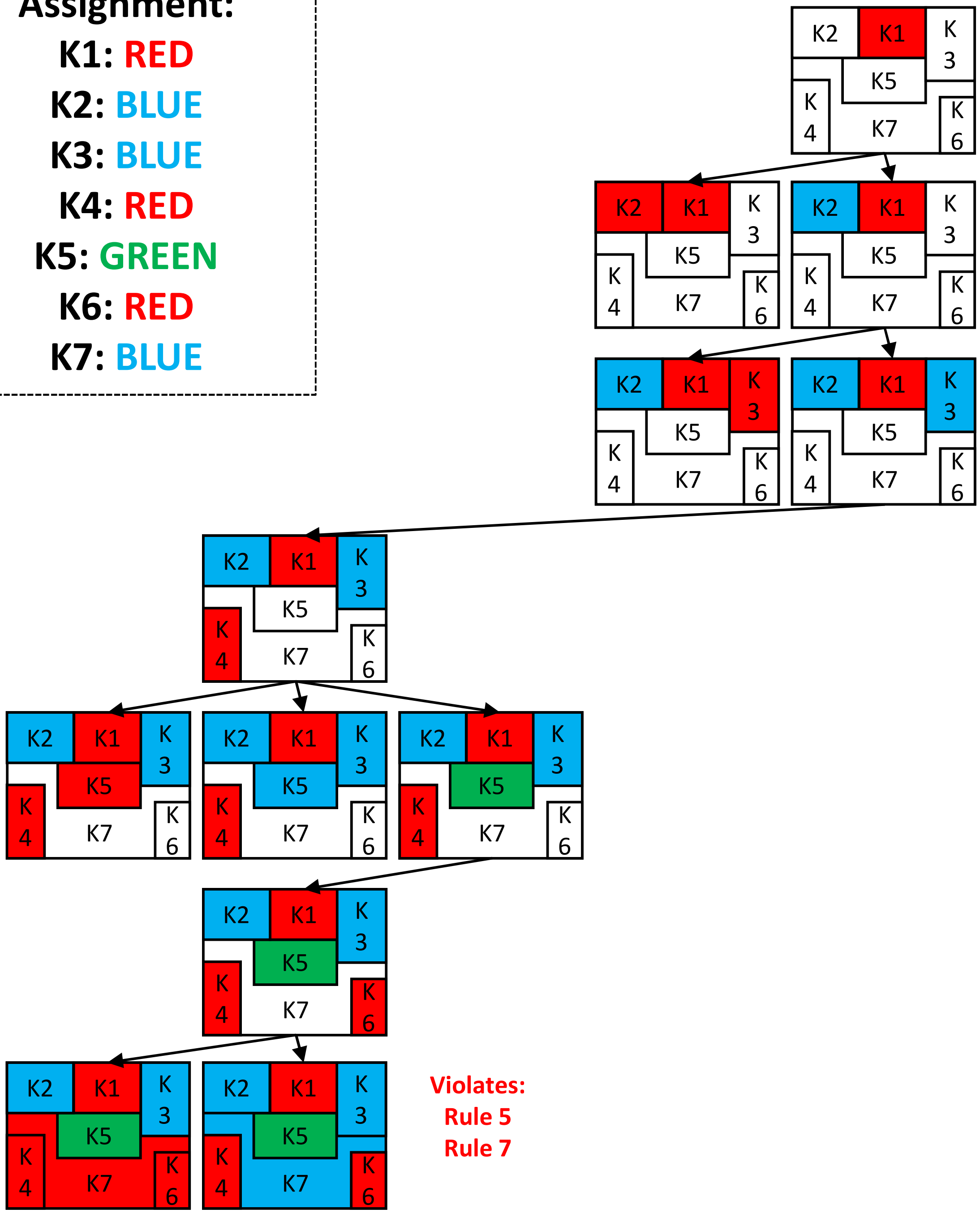
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: RED
K7: BLUE



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

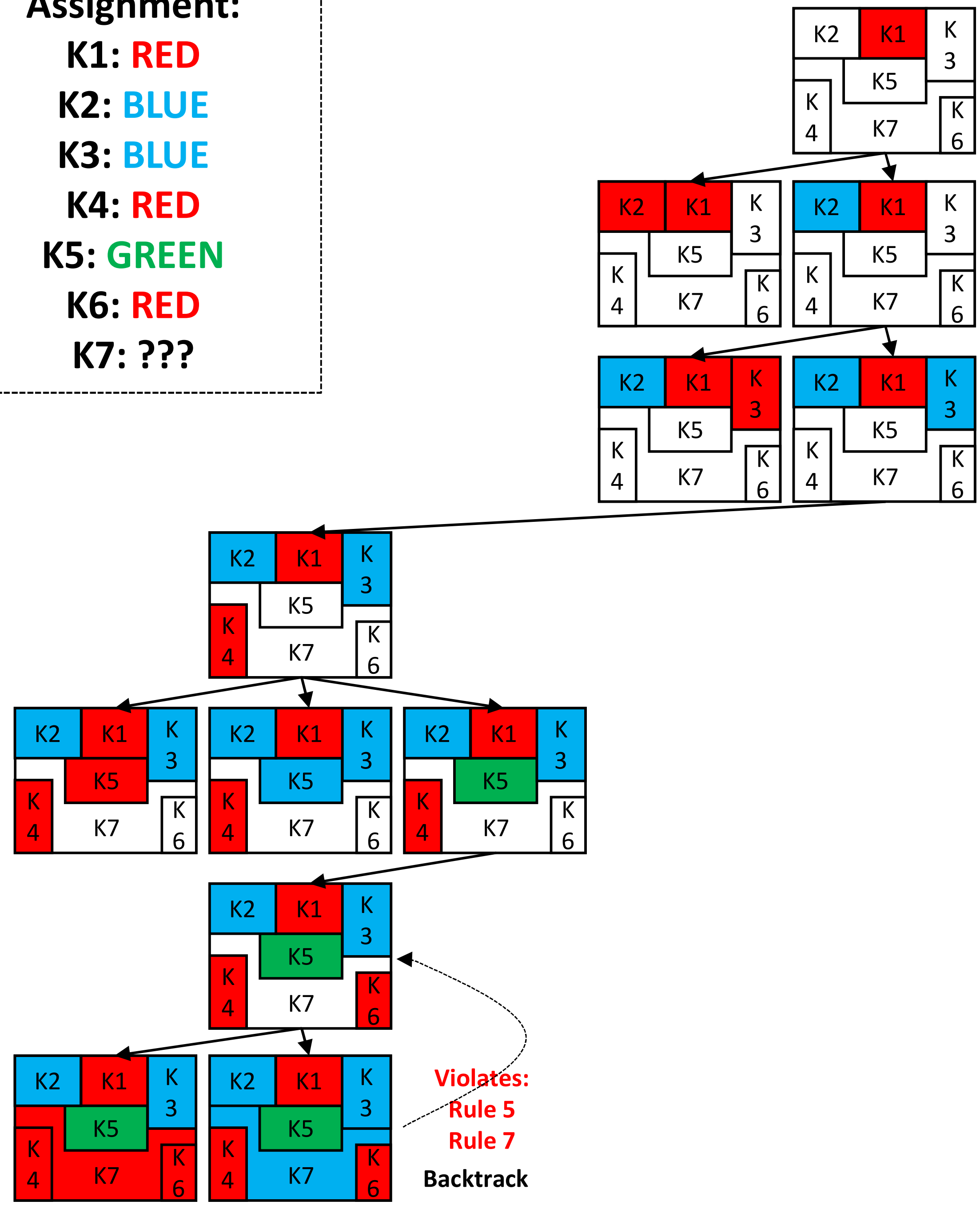
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: RED
K7: BLUE



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

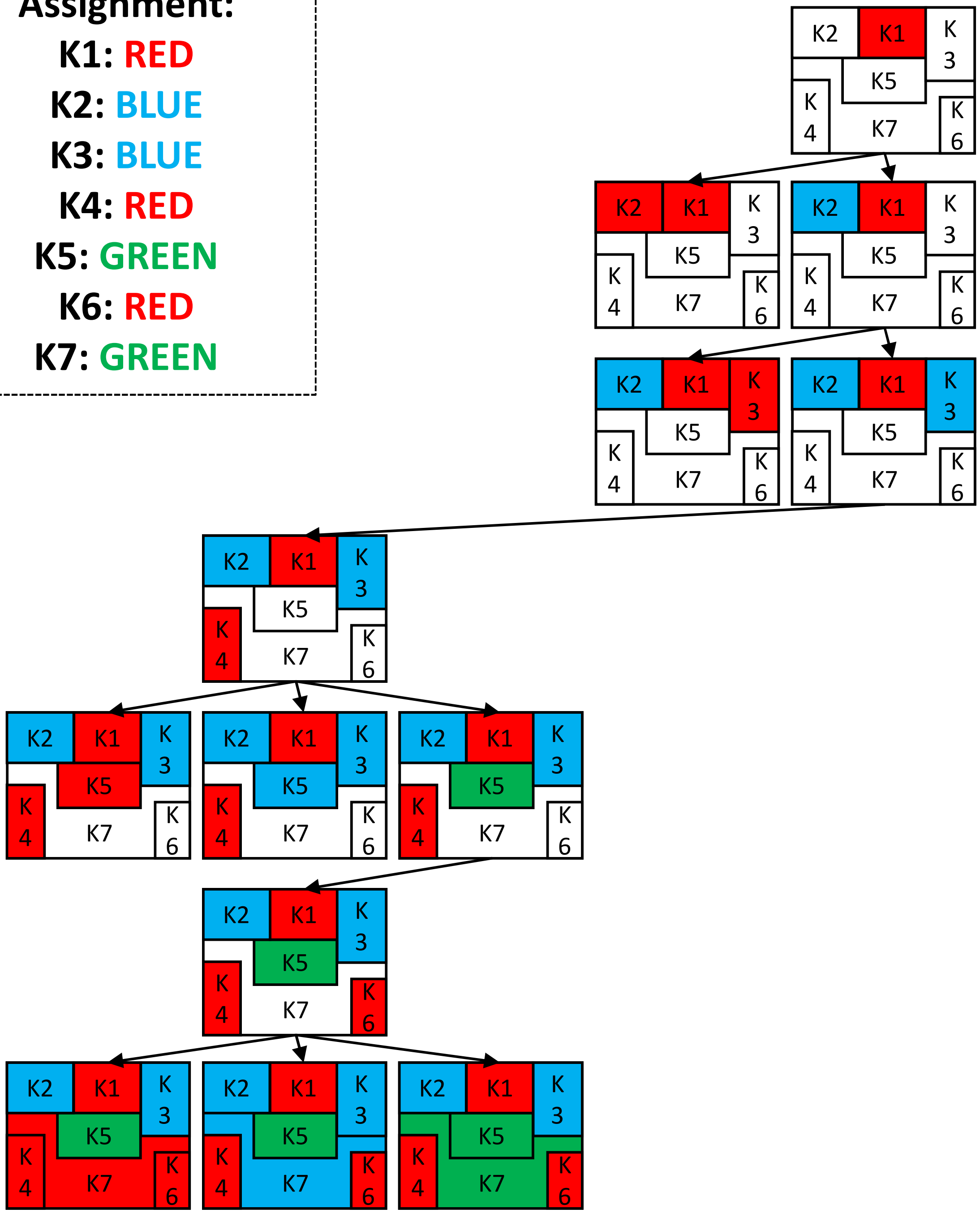
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: RED
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

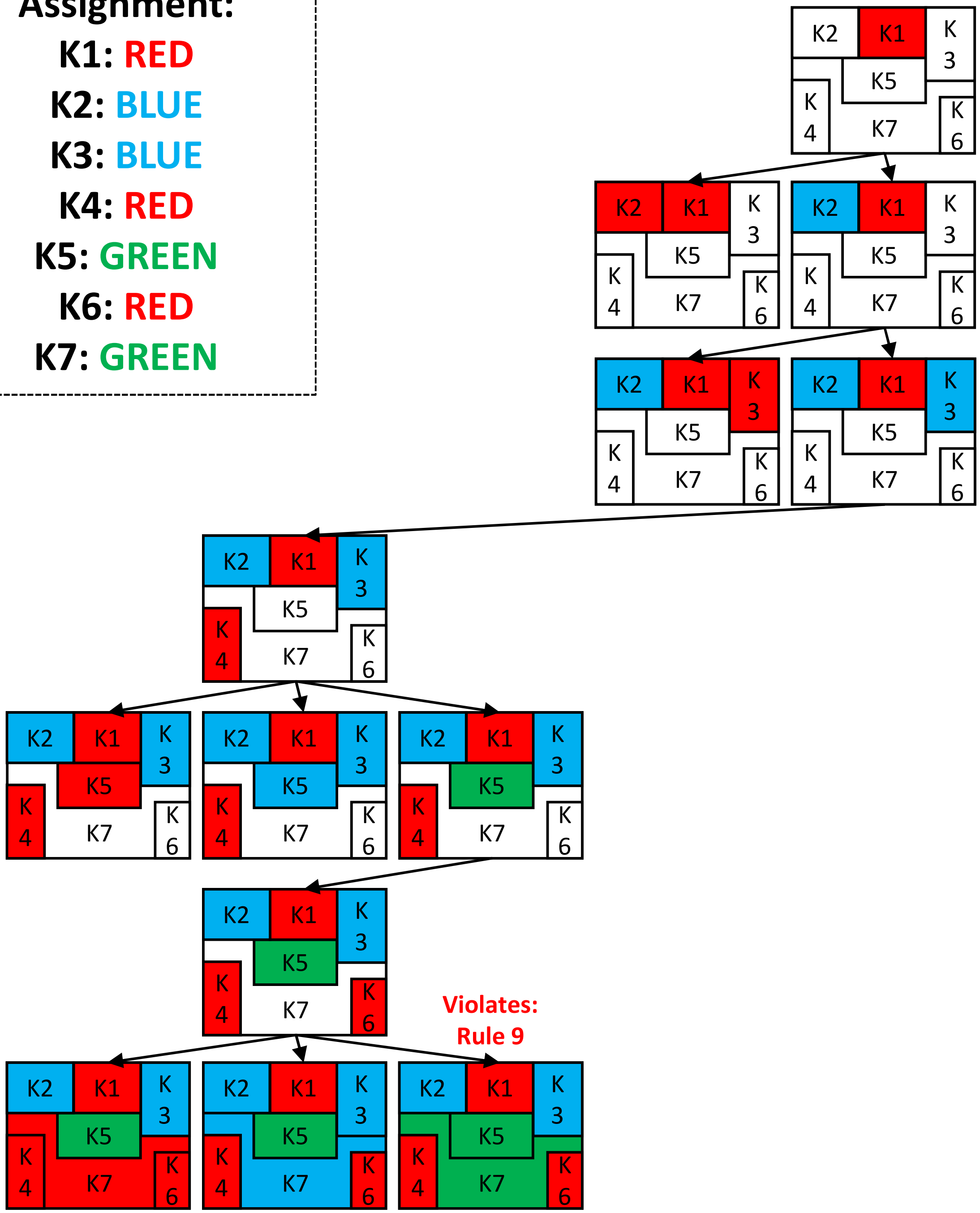
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: RED
K7: GREEN



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: RED
K7: GREEN



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:

K1: RED

K2: BLUE

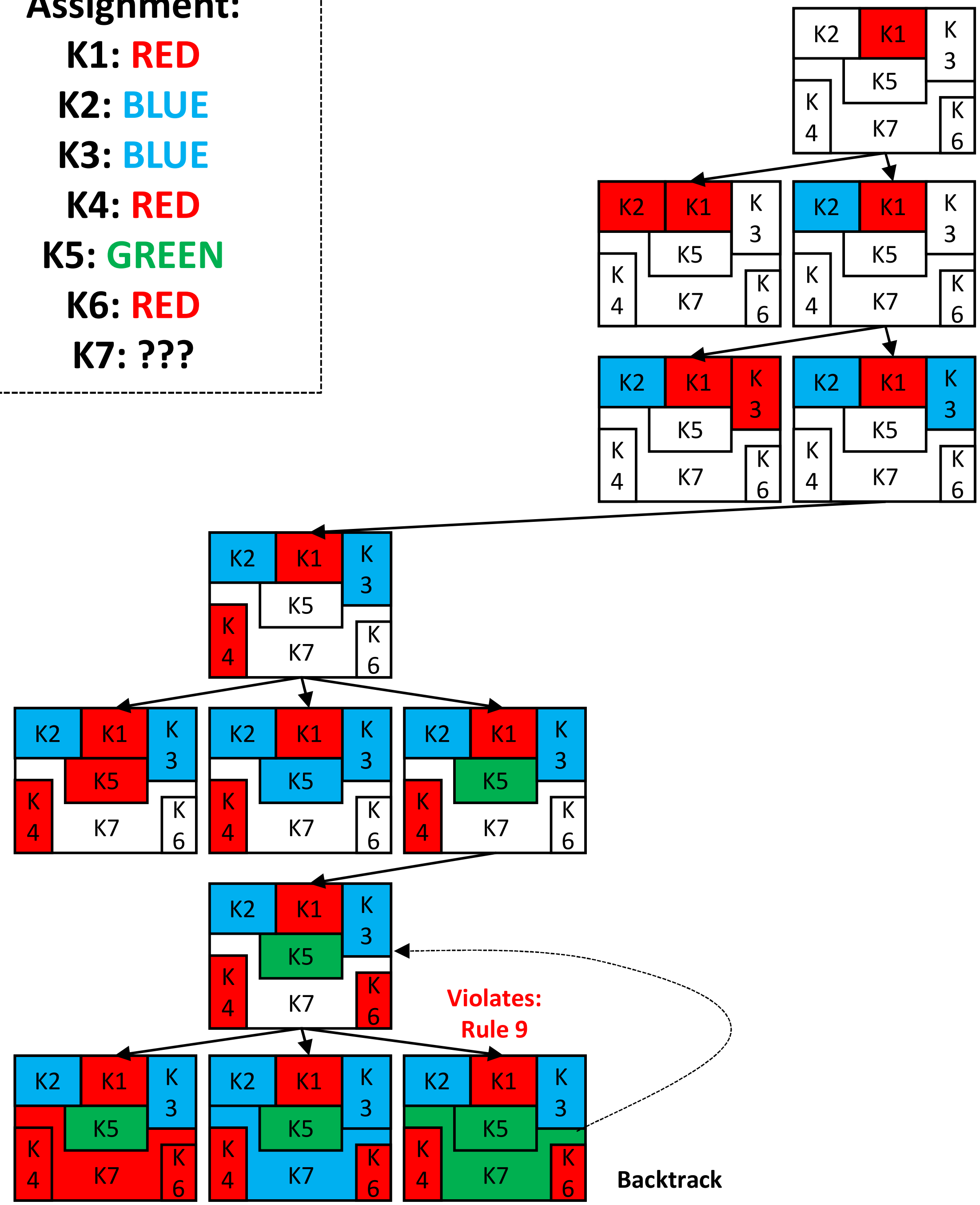
K3: BLUE

K4: RED

K5: GREEN

K6: RED

K7: ???



Constraints:

Rule 1: K1 ≠ K2

Rule 2: K1 ≠ K3

Rule 3: K1 ≠ K5

Rule 4: K2 ≠ K5

Rule 5: K2 ≠ K7

Rule 6: K3 ≠ K5

Rule 7: K3 ≠ K7

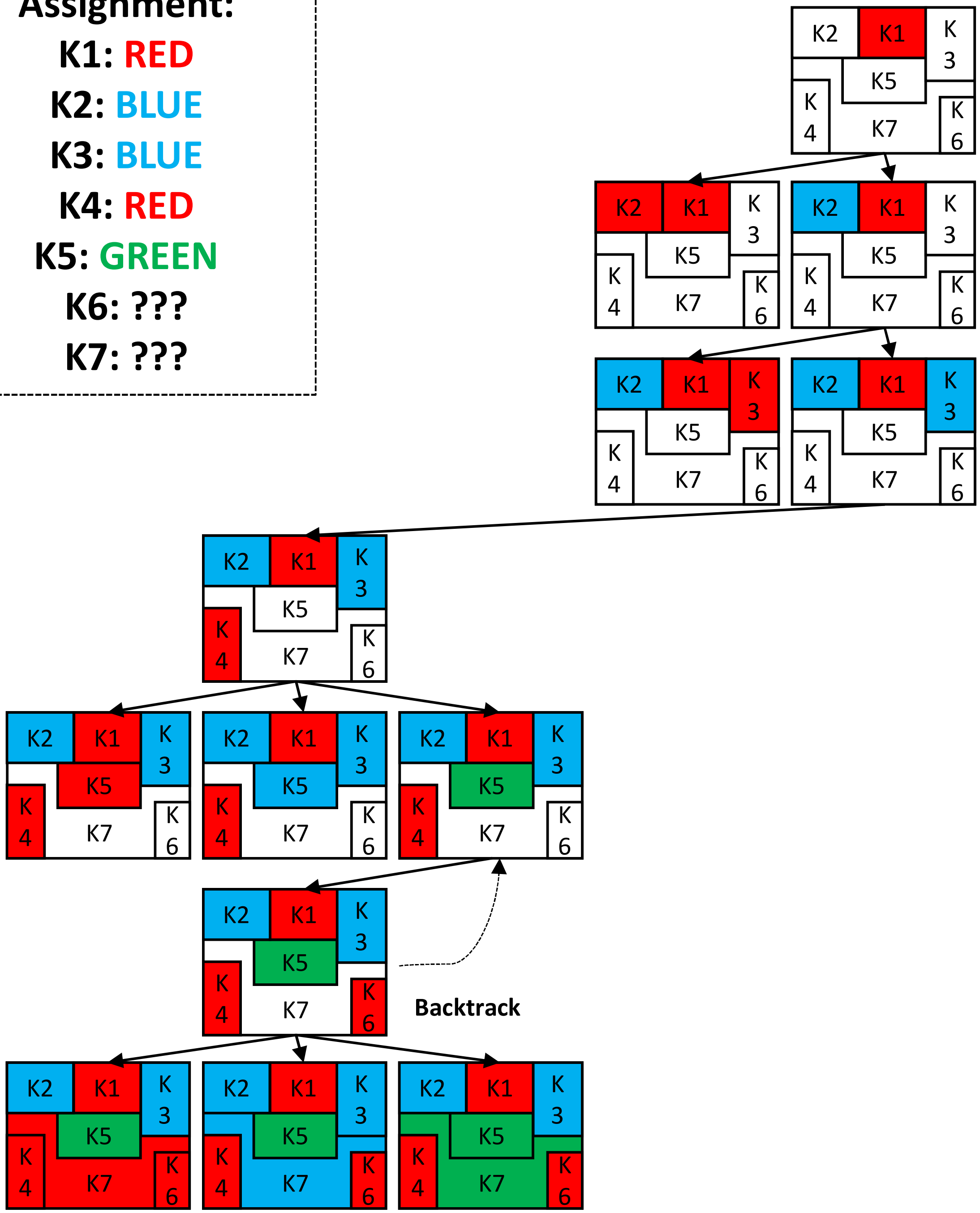
Rule 8: K4 ≠ K7

Rule 9: K5 ≠ K7

Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

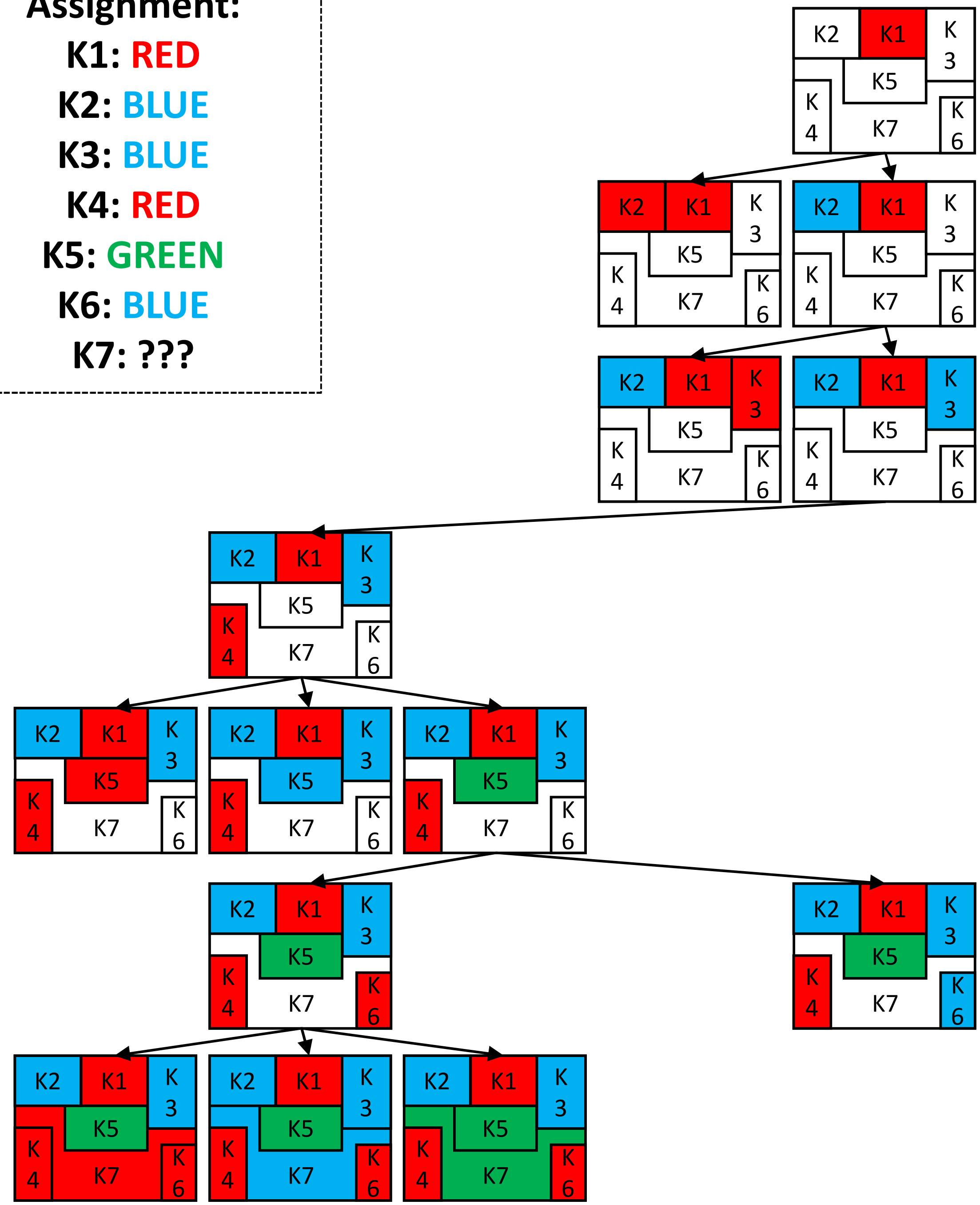
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order:** RED, BLUE, GREEN

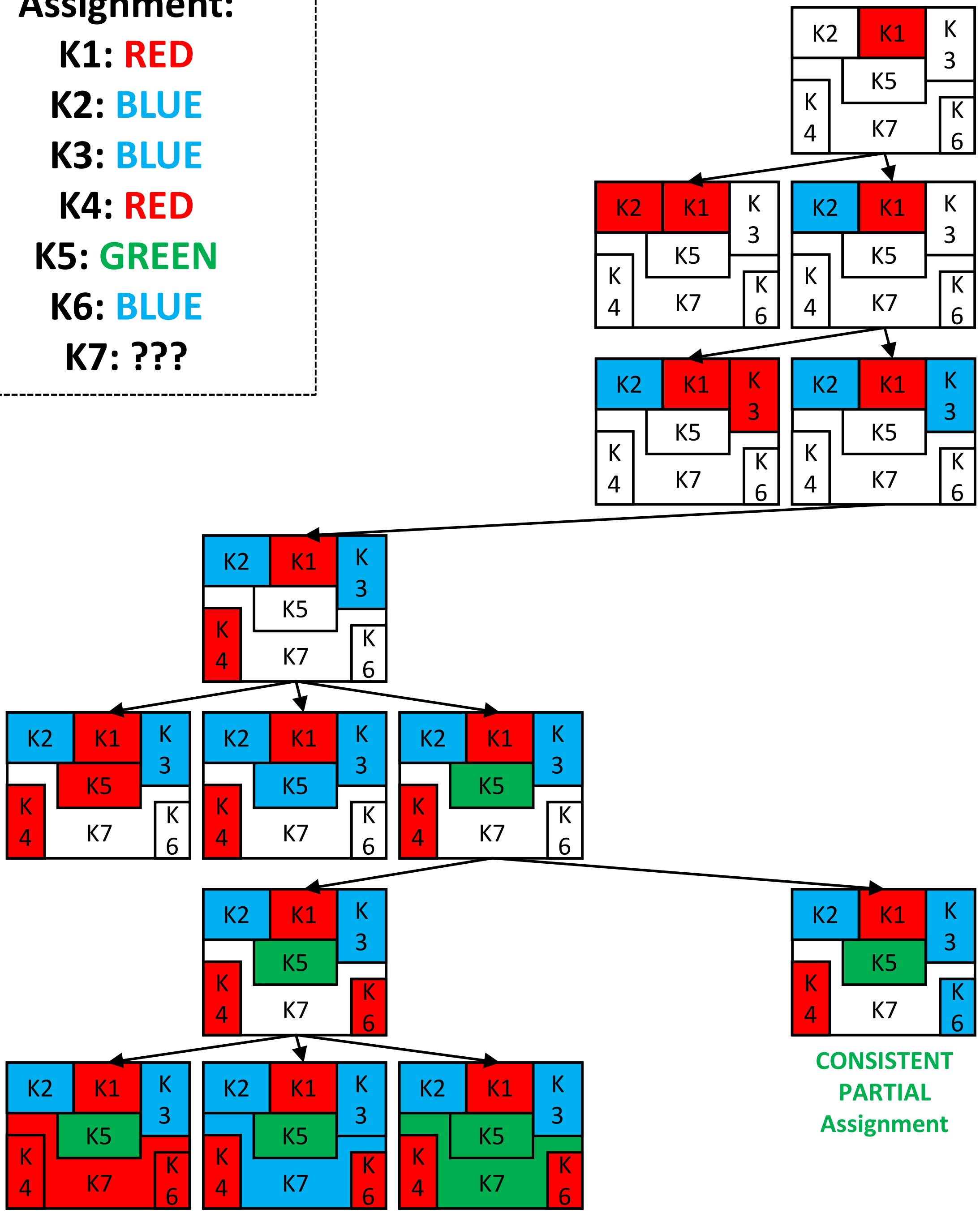
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: BLUE
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

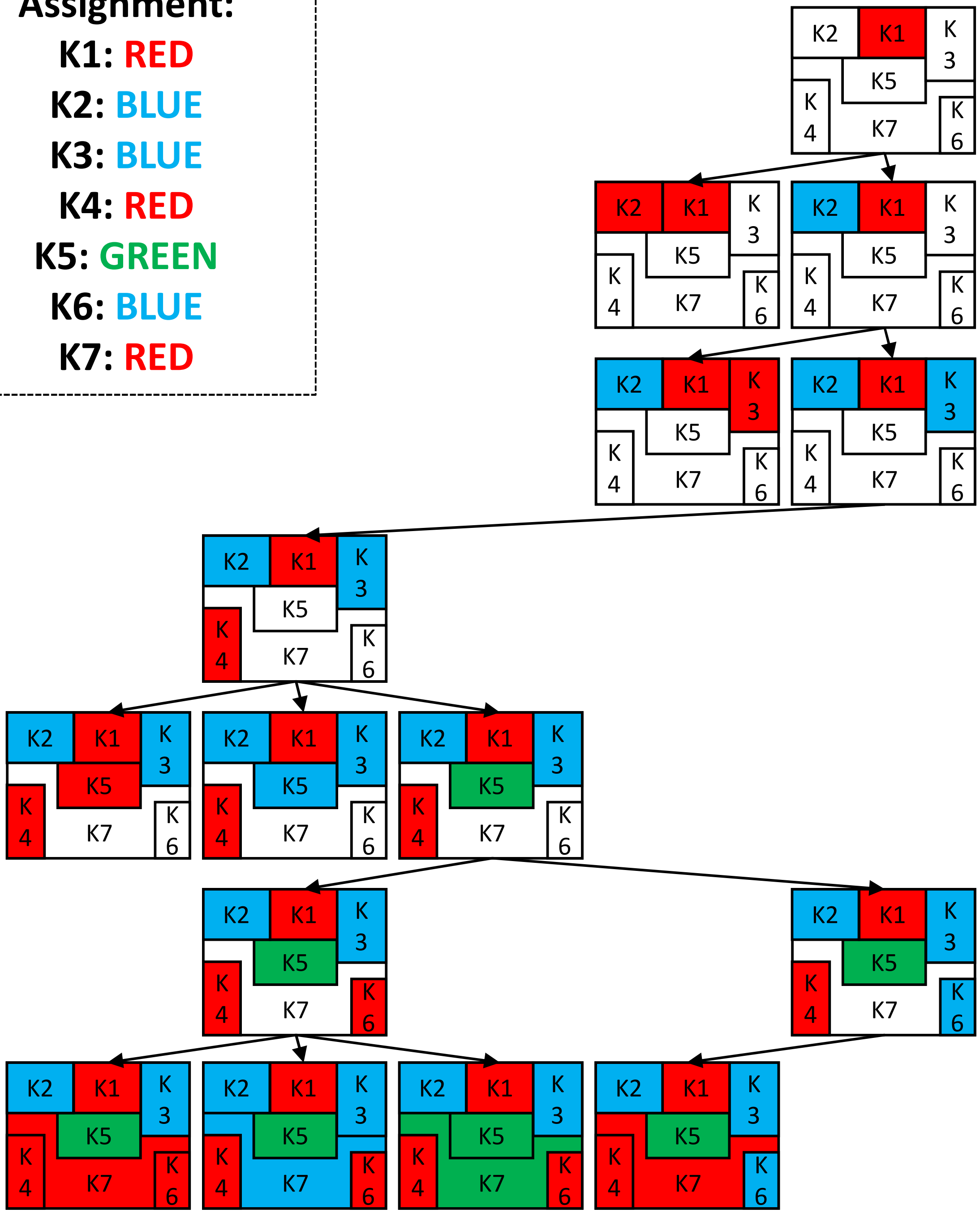
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: BLUE
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

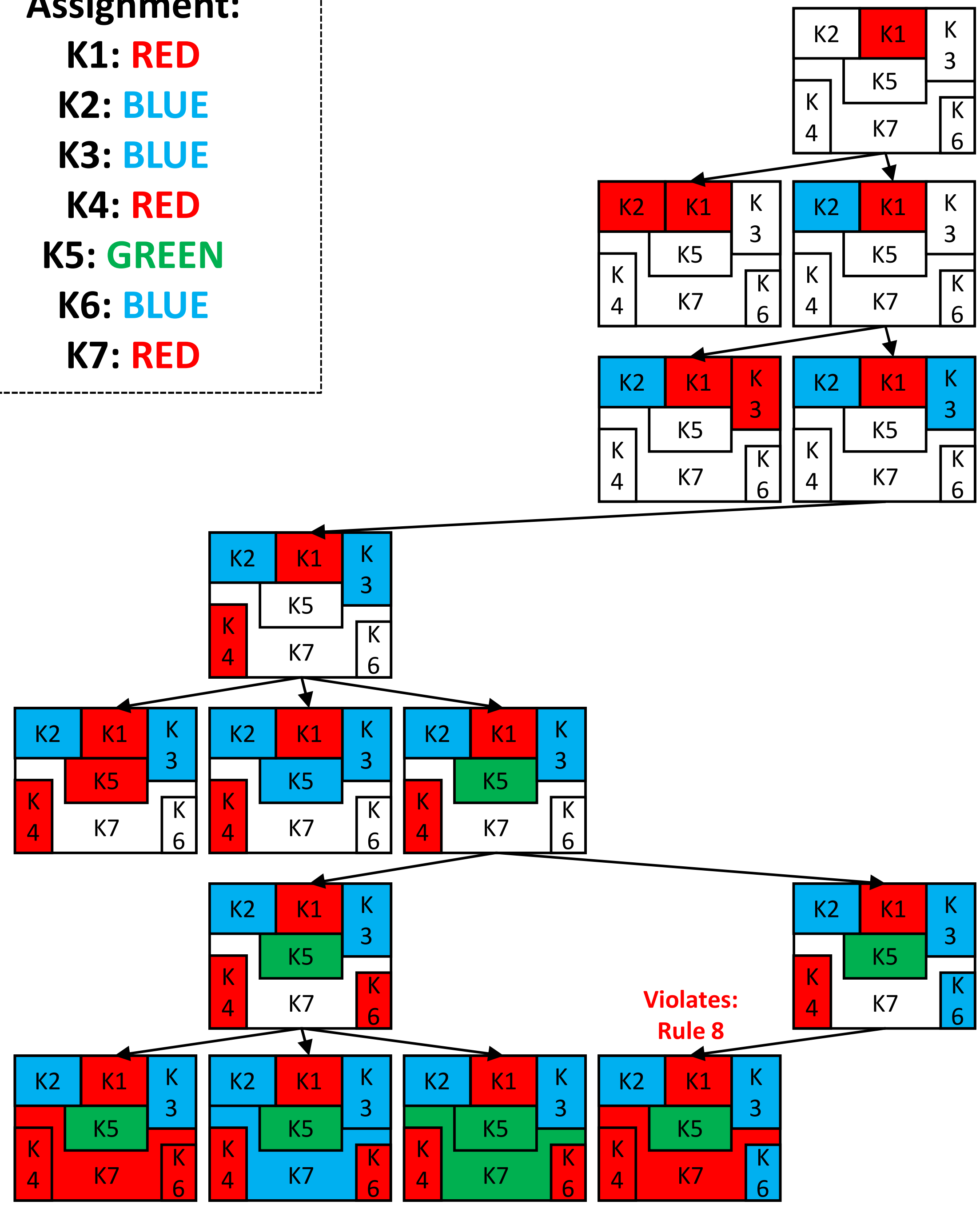
Assignment:
K1: **RED**
K2: **BLUE**
K3: **BLUE**
K4: **RED**
K5: **GREEN**
K6: **BLUE**
K7: **RED**



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

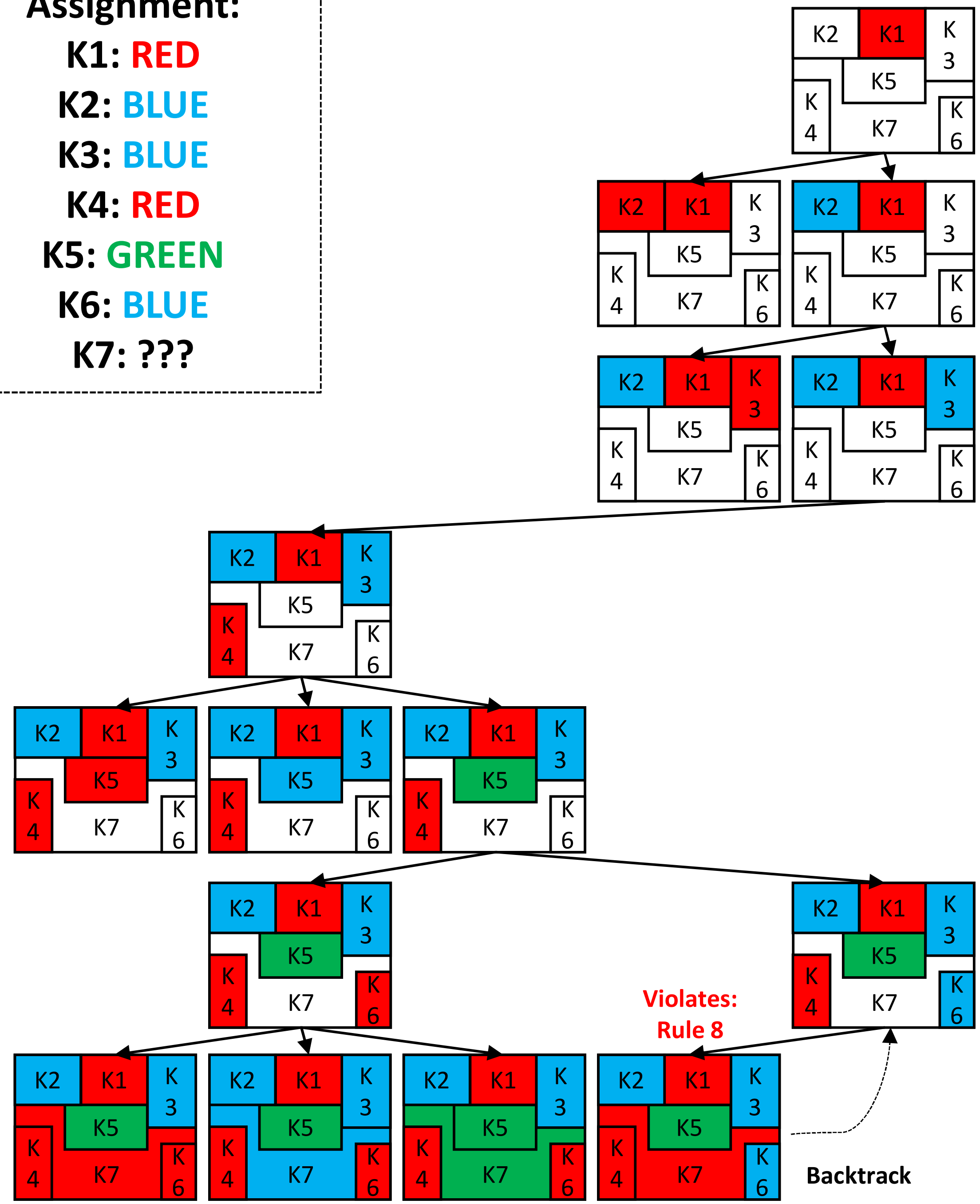
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: BLUE
K7: RED



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: BLUE
K7: ???

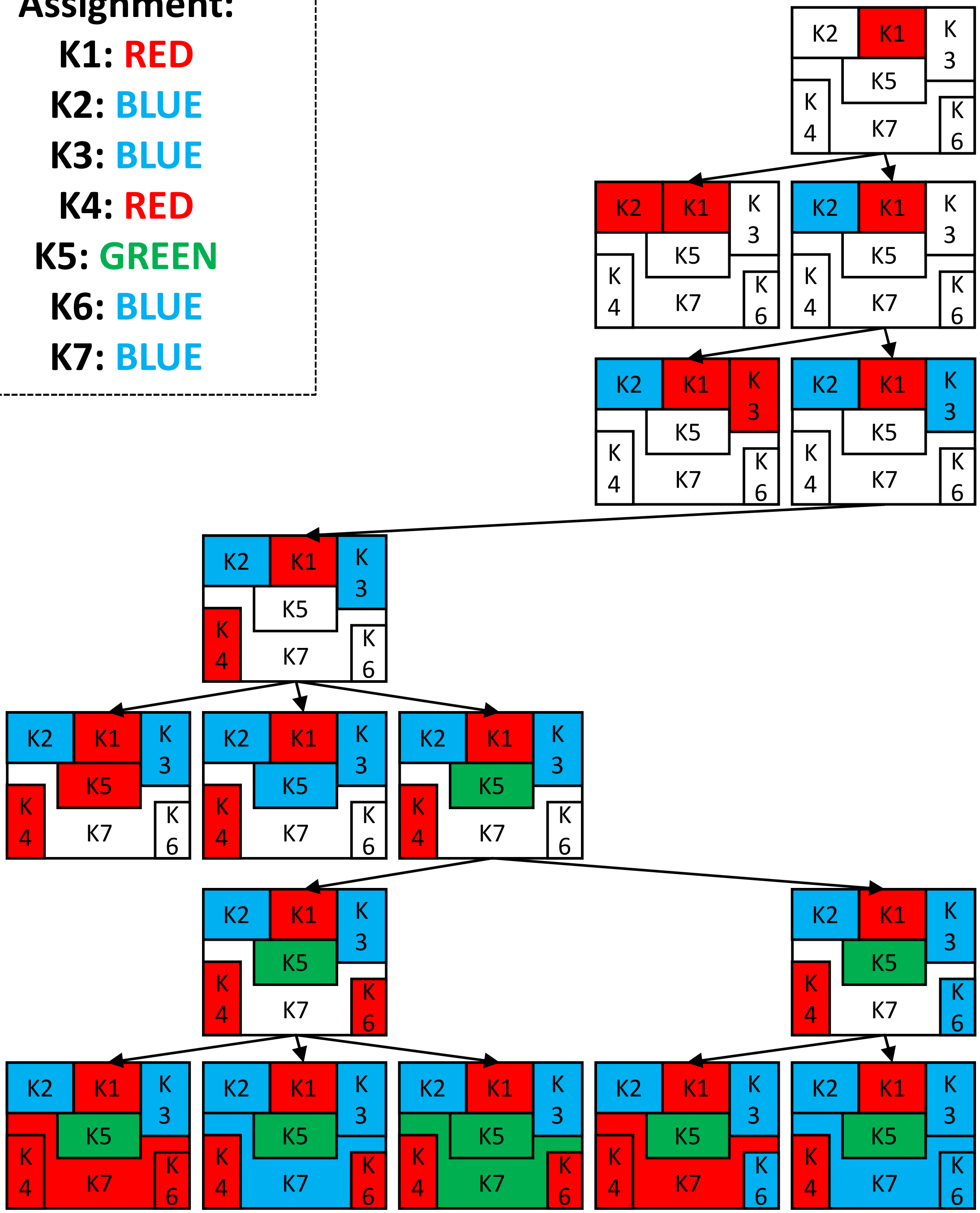


Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: BLUE
K7: BLUE

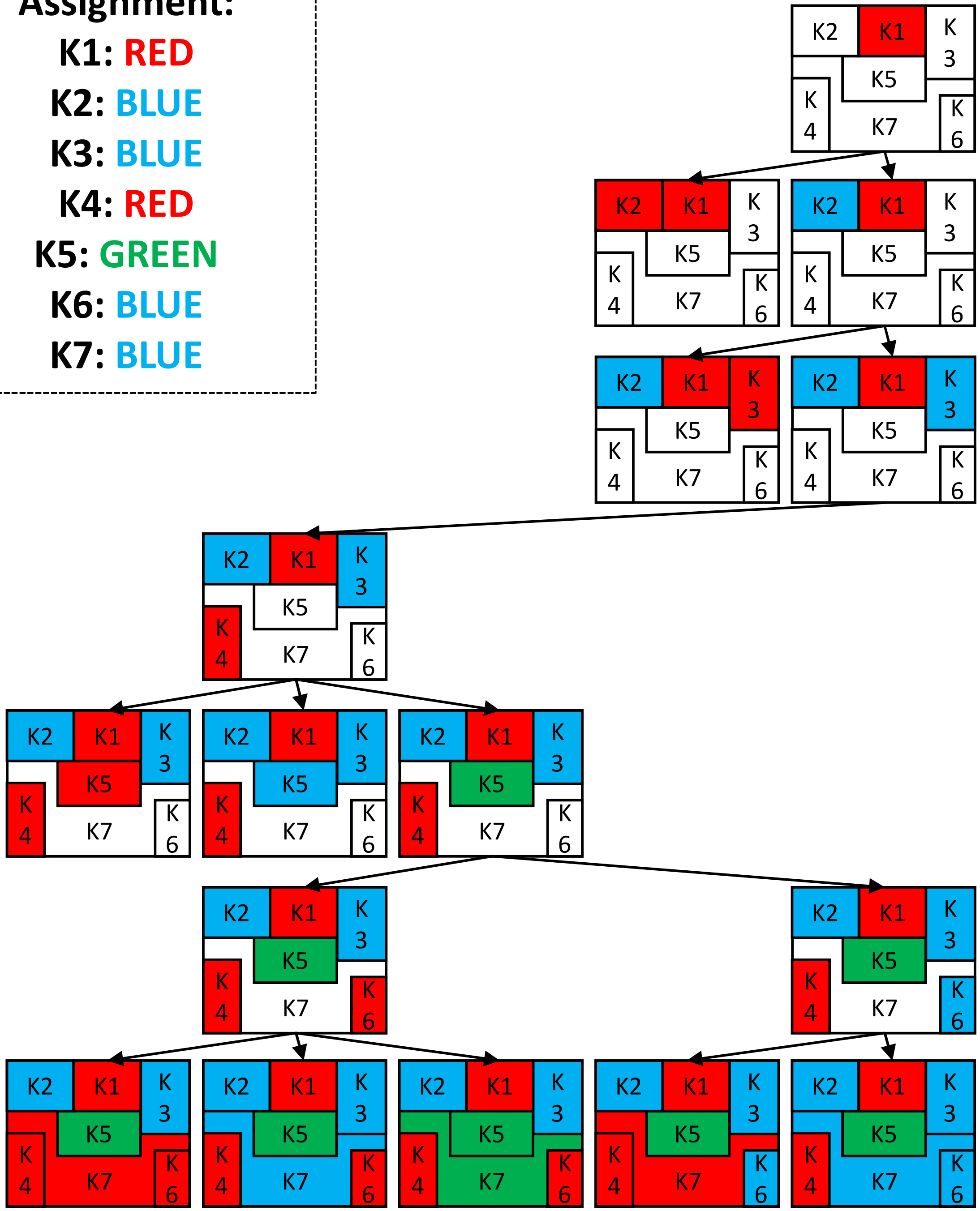
Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: **RED**
K2: **BLUE**
K3: **BLUE**
K4: **RED**
K5: **GREEN**
K6: **BLUE**
K7: **BLUE**

Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Violates:
Rule 5
Rule 7
Rule 10

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:

K1: RED

K2: BLUE

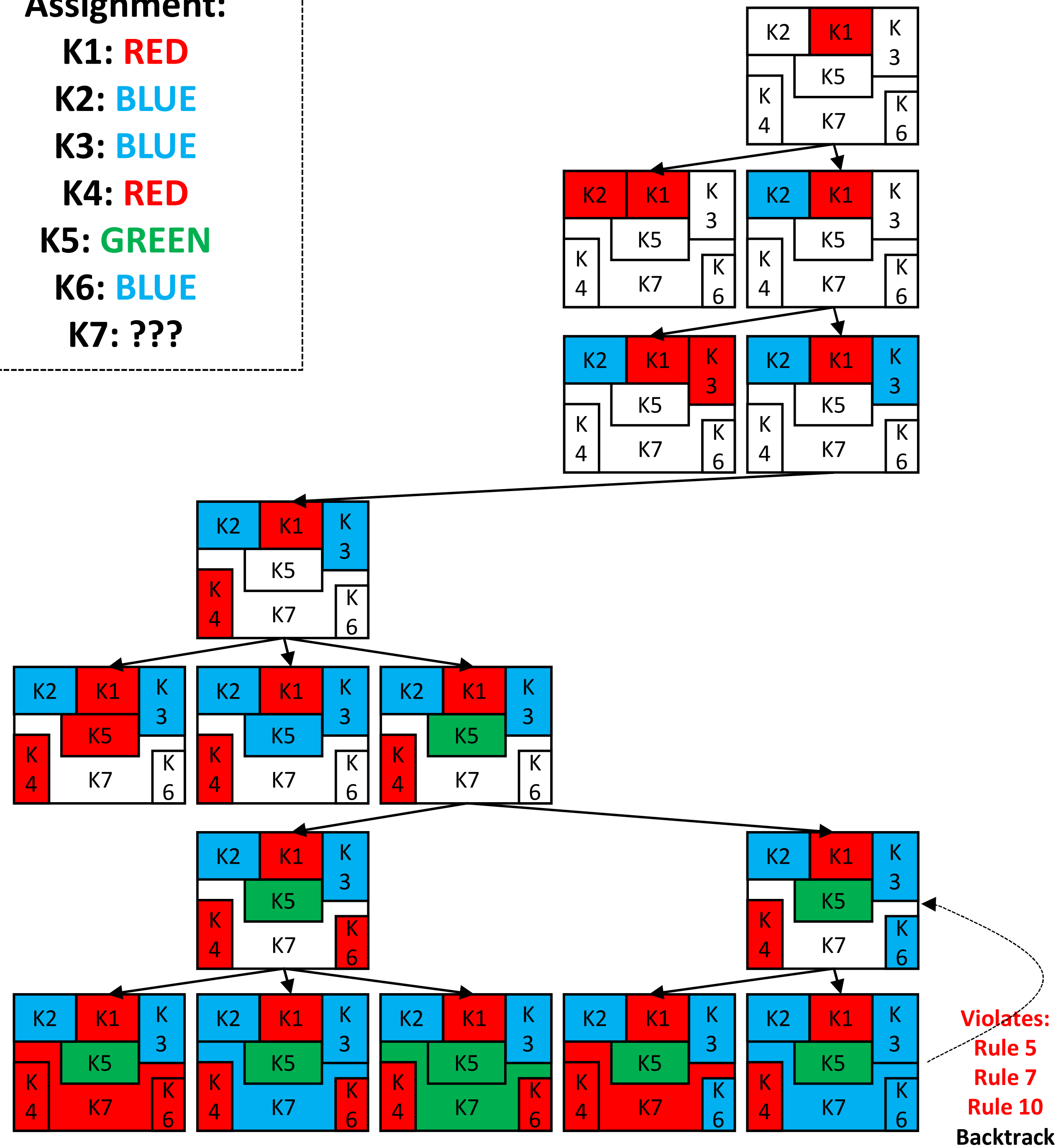
K3: BLUE

K4: RED

K5: GREEN

K6: BLUE

K7: ???



Constraints:

Rule 1: K1 ≠ K2

Rule 2: K1 ≠ K3

Rule 3: K1 ≠ K5

Rule 4: K2 ≠ K5

Rule 5: K2 ≠ K7

Rule 6: K3 ≠ K5

Rule 7: K3 ≠ K7

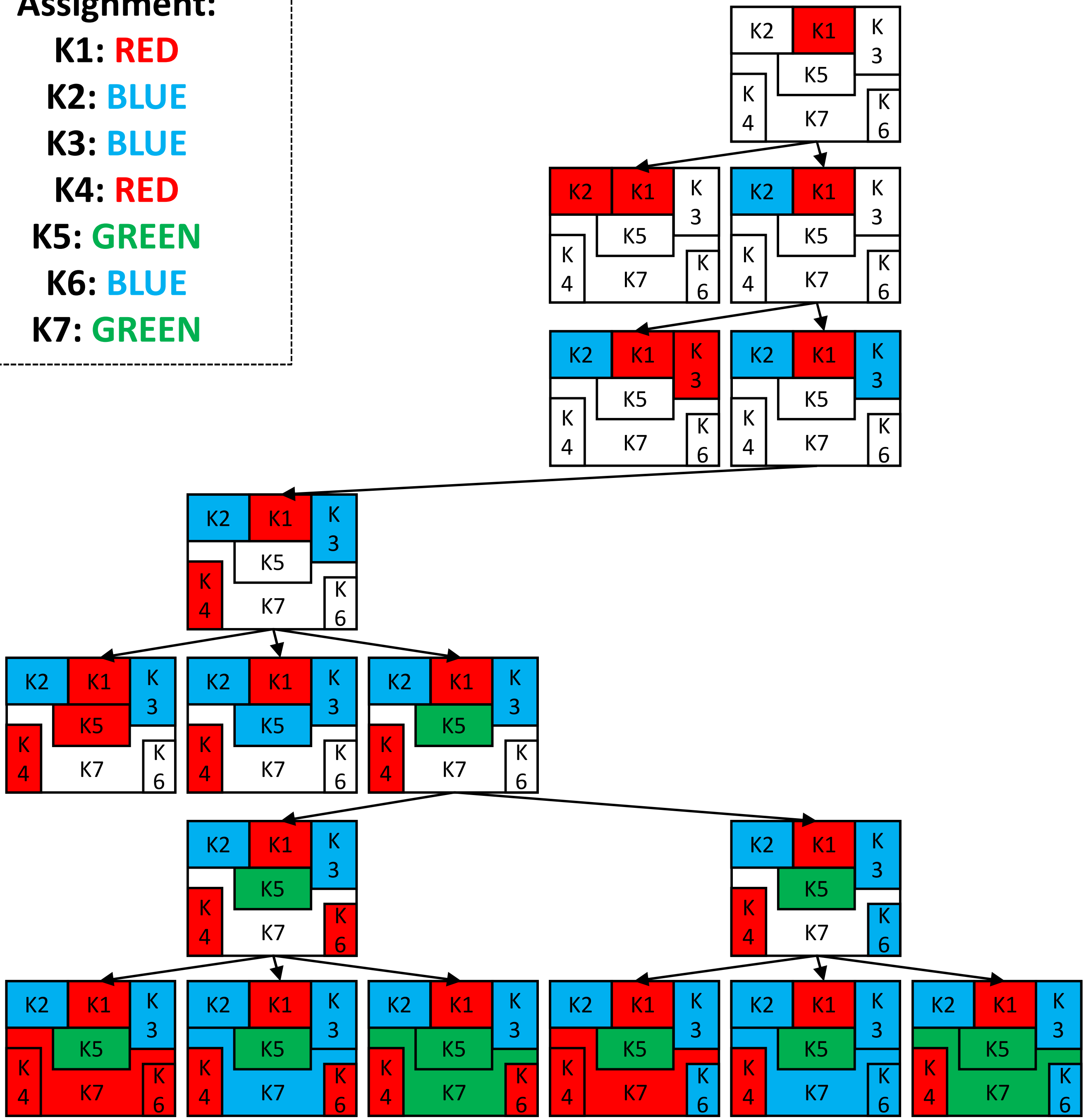
Rule 8: K4 ≠ K7

Rule 9: K5 ≠ K7

Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order:** RED, BLUE, GREEN

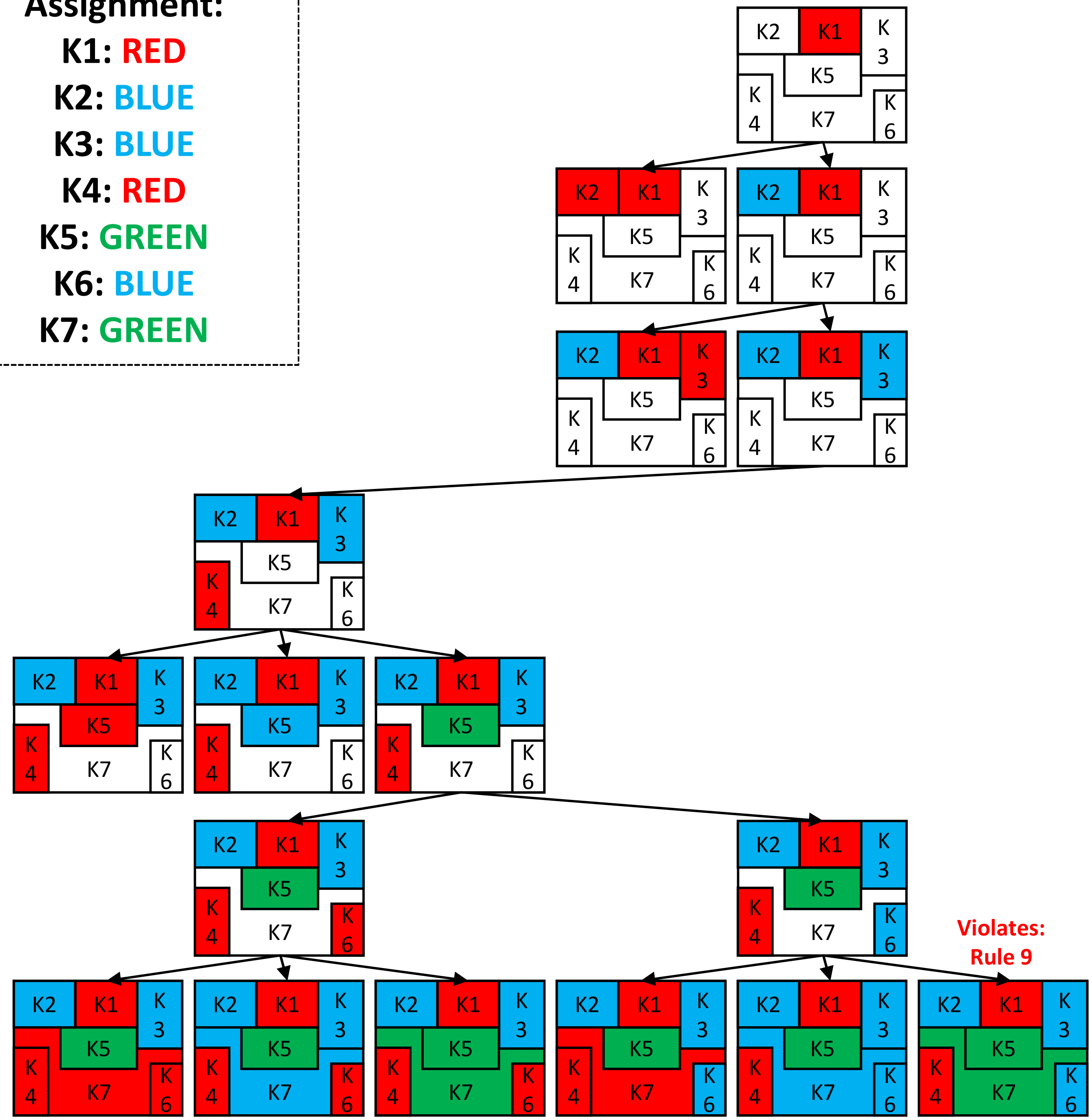
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: BLUE
K7: GREEN



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: BLUE
K7: GREEN



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:

K1: RED

K2: BLUE

K3: BLUE

K4: RED

K5: GREEN

K6: BLUE

K7: ???

Constraints:

Rule 1: K1 ≠ K2

Rule 2: K1 ≠ K3

Rule 3: K1 ≠ K5

Rule 4: K2 ≠ K5

Rule 5: K2 ≠ K7

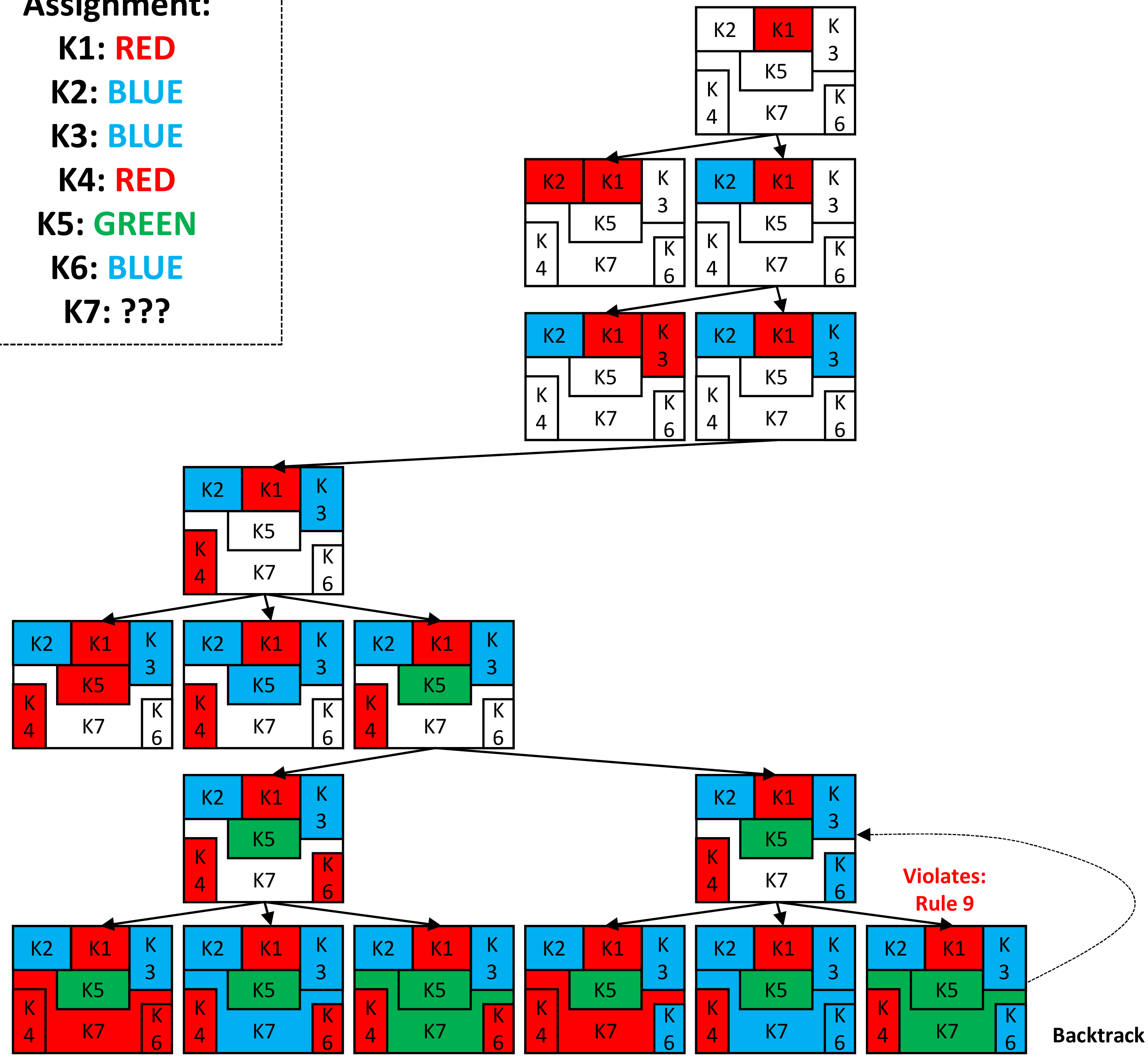
Rule 6: K3 ≠ K5

Rule 7: K3 ≠ K7

Rule 8: K4 ≠ K7

Rule 9: K5 ≠ K7

Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order:** RED, BLUE, GREEN

Assignment:

K1: RED

K2: BLUE

K3: BLUE

K4: RED

K5: GREEN

K6: ???

K7: ???

Constraints:

Rule 1: K1 ≠ K2

Rule 2: K1 ≠ K3

Rule 3: K1 ≠ K5

Rule 4: K2 ≠ K5

Rule 5: K2 ≠ K7

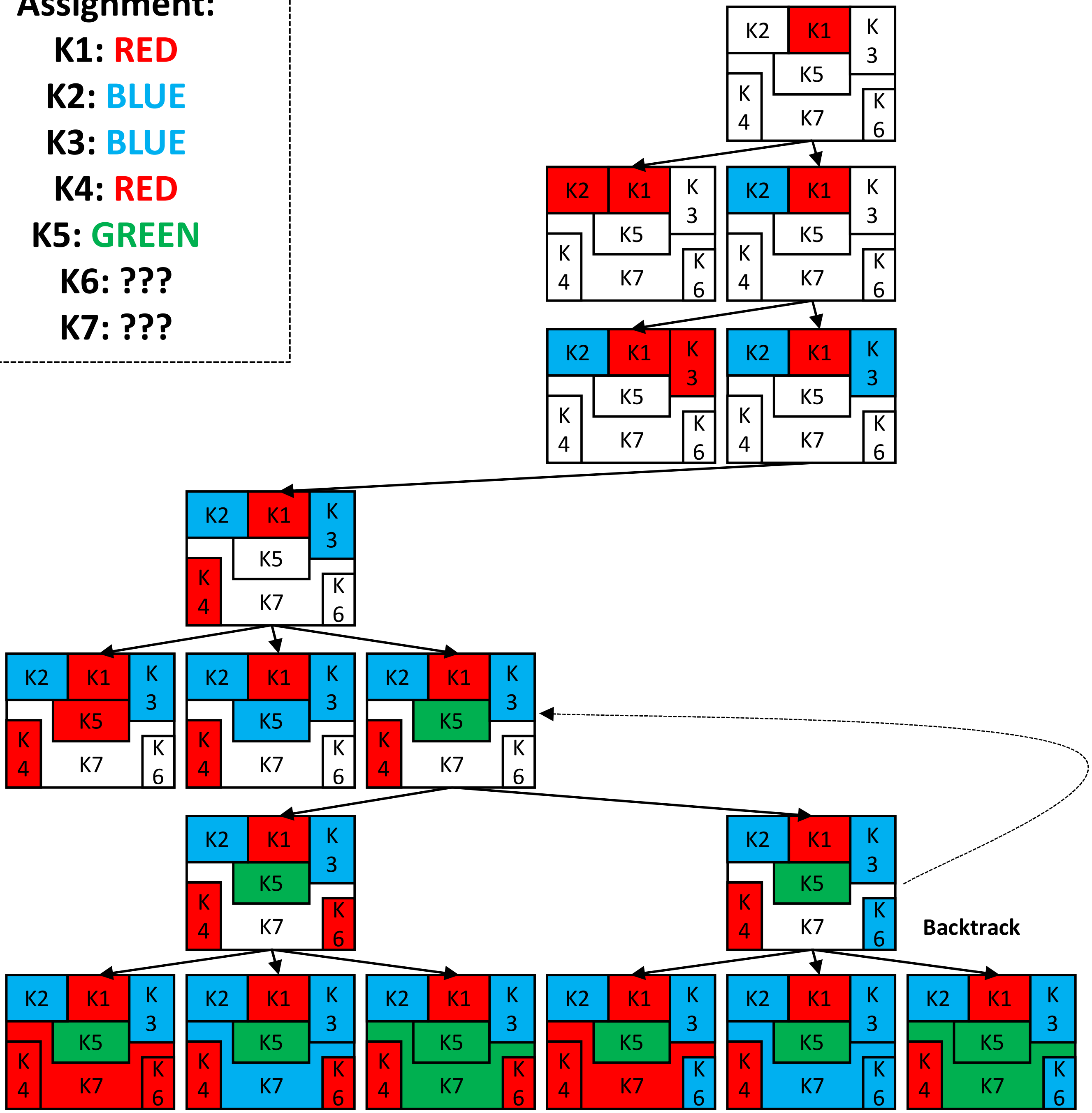
Rule 6: K3 ≠ K5

Rule 7: K3 ≠ K7

Rule 8: K4 ≠ K7

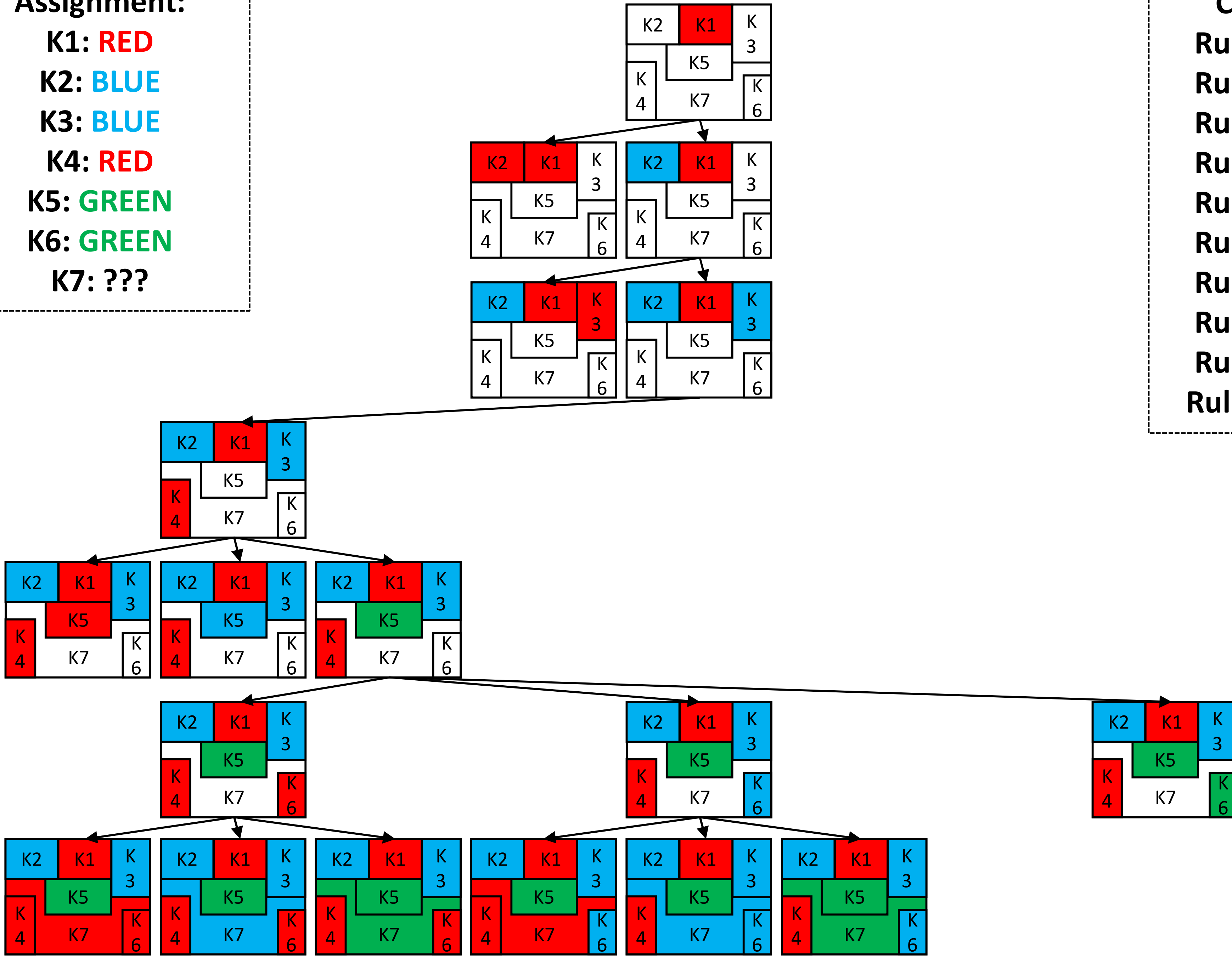
Rule 9: K5 ≠ K7

Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order:** RED, BLUE, GREEN

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: GREEN
K7: ???

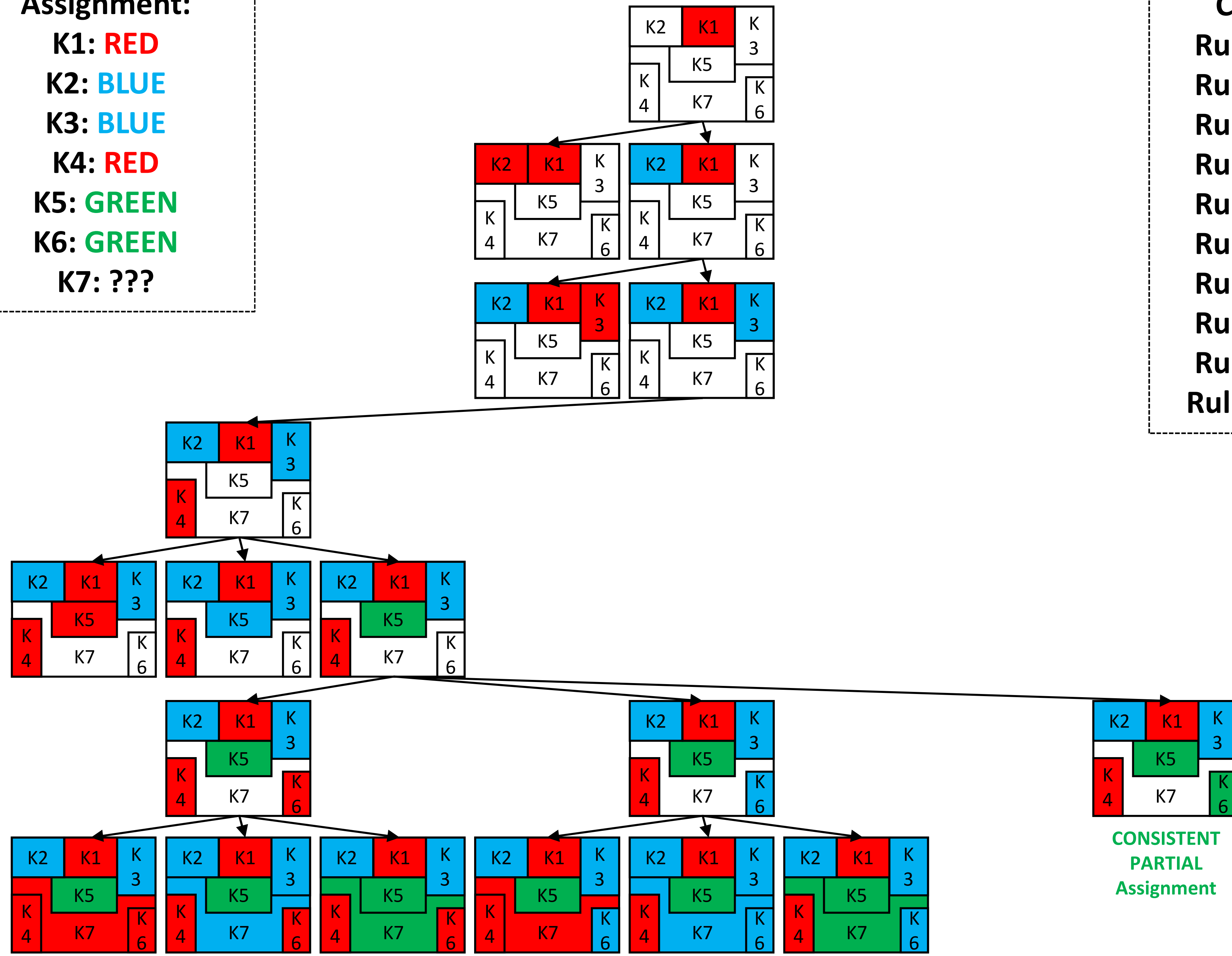


Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

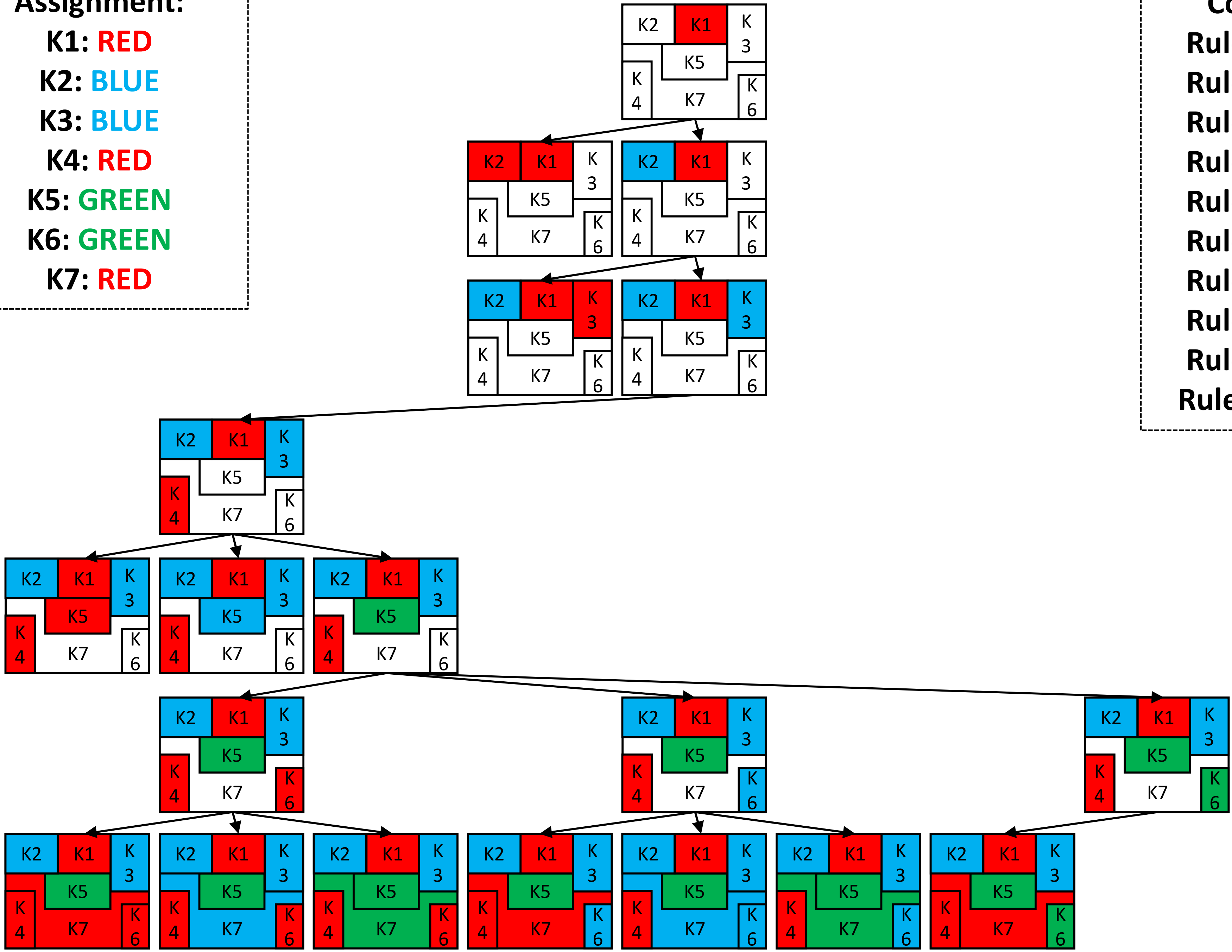
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: GREEN
K7: ???

Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order:** RED, BLUE, GREEN

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: GREEN
K7: RED

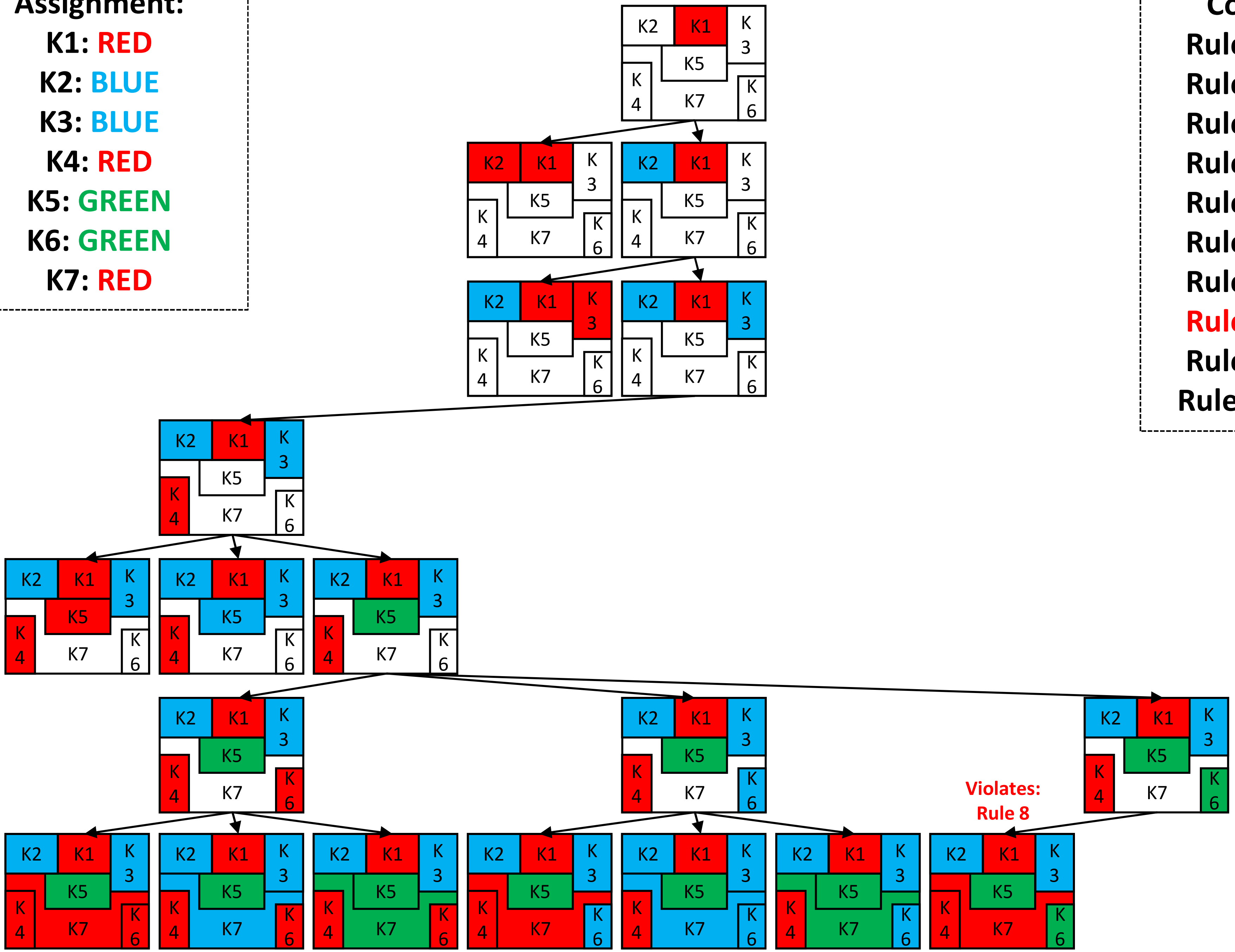


Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

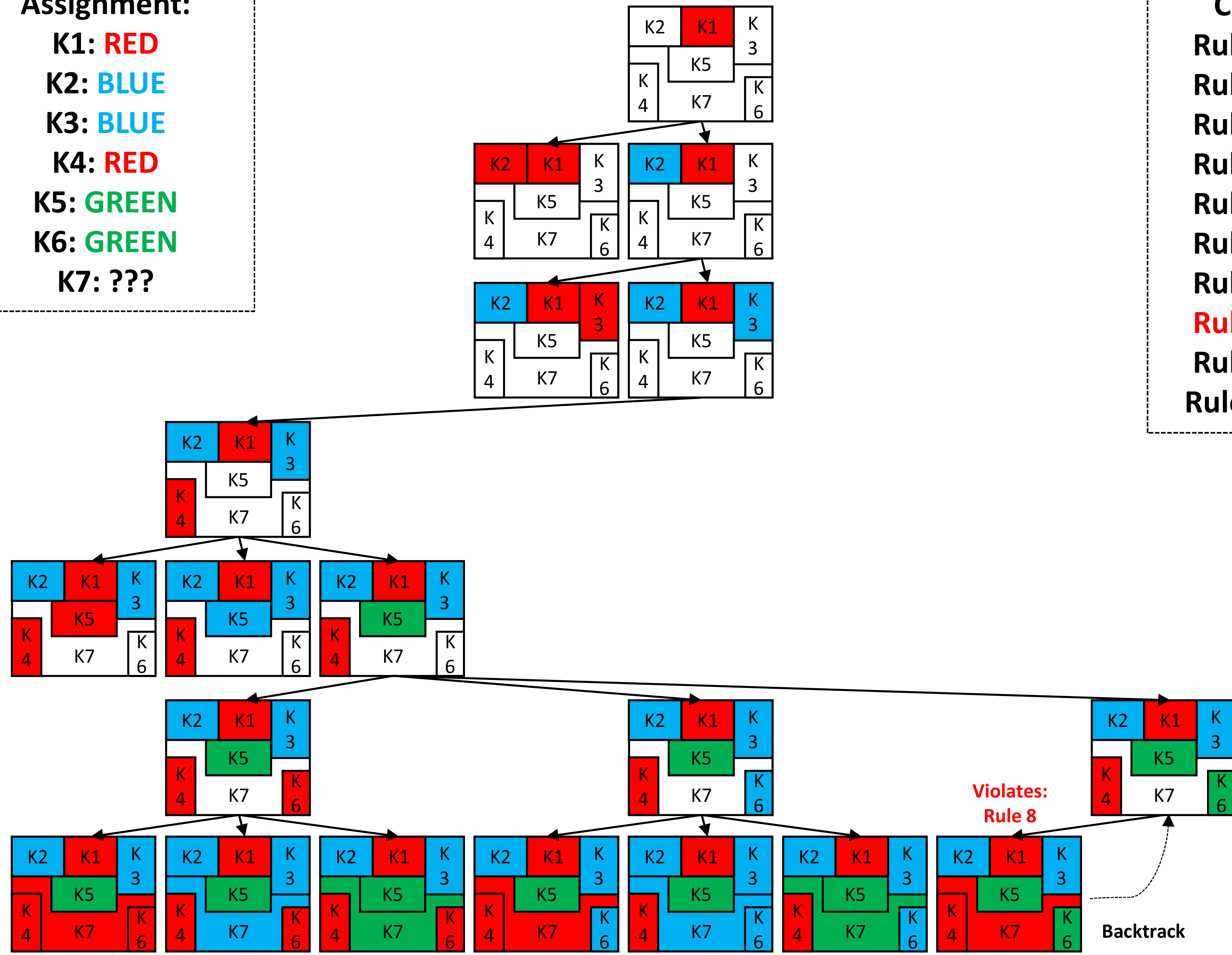
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: GREEN
K7: RED

Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

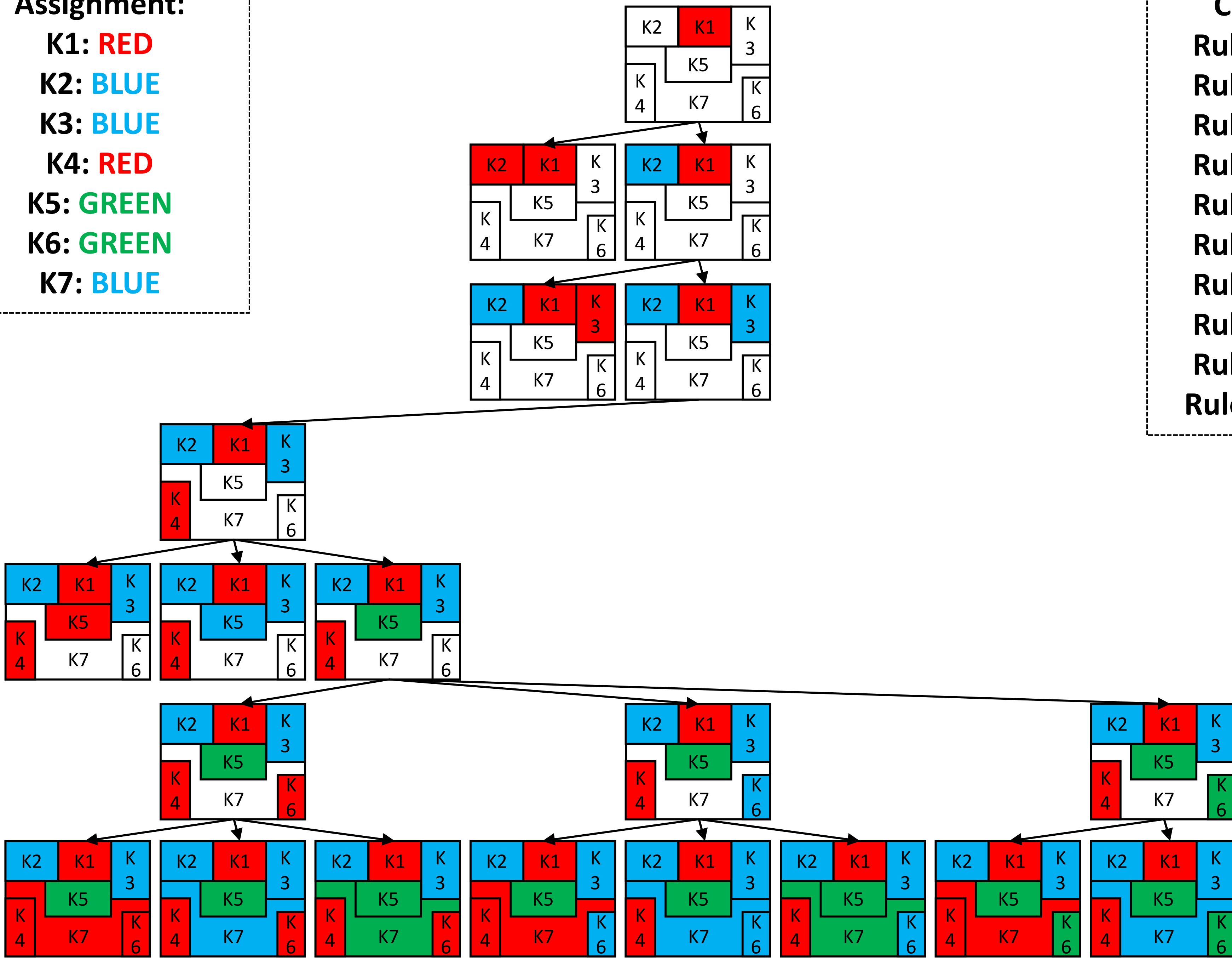
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: GREEN
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

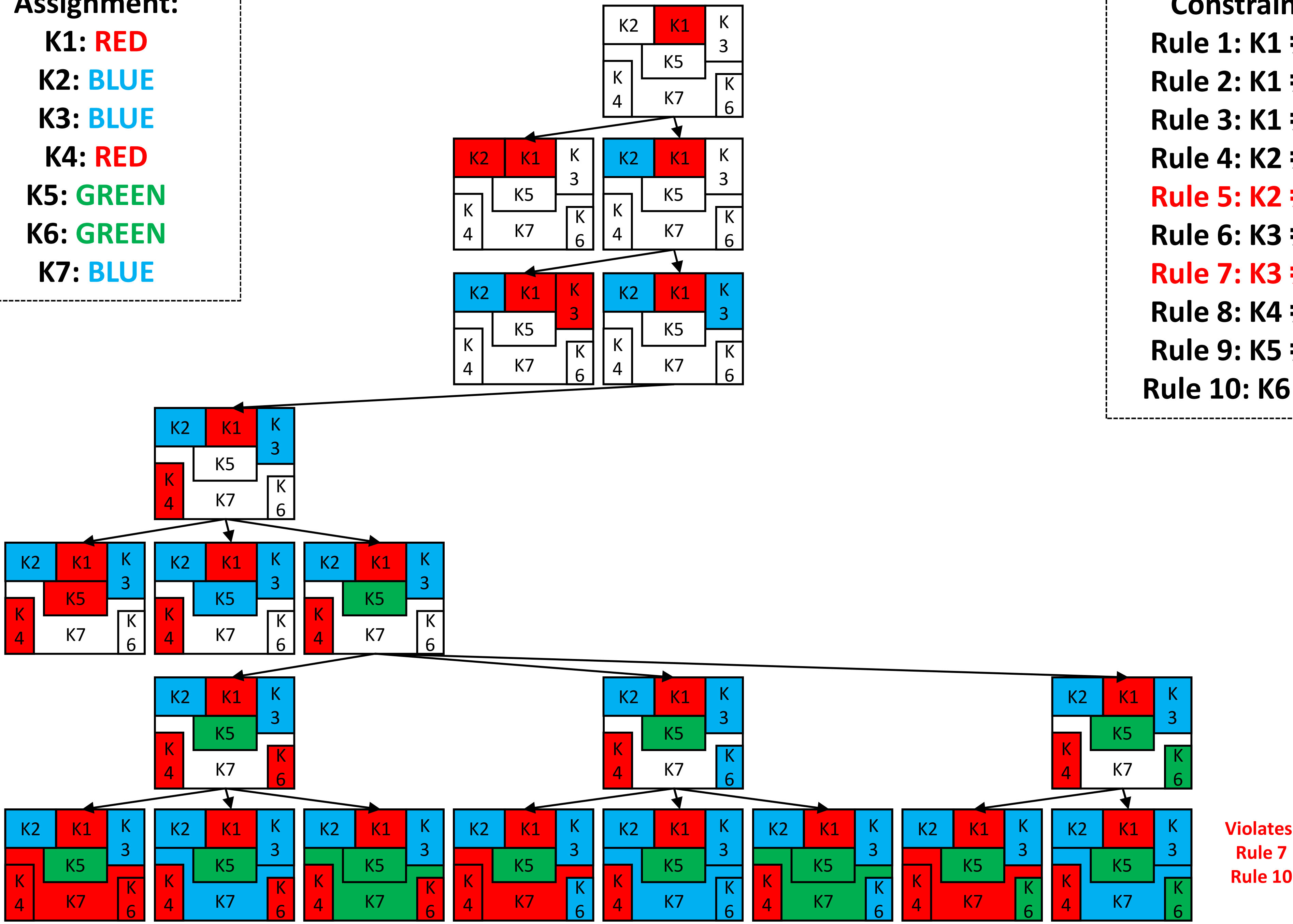
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: GREEN
K7: BLUE



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: GREEN
K7: BLUE

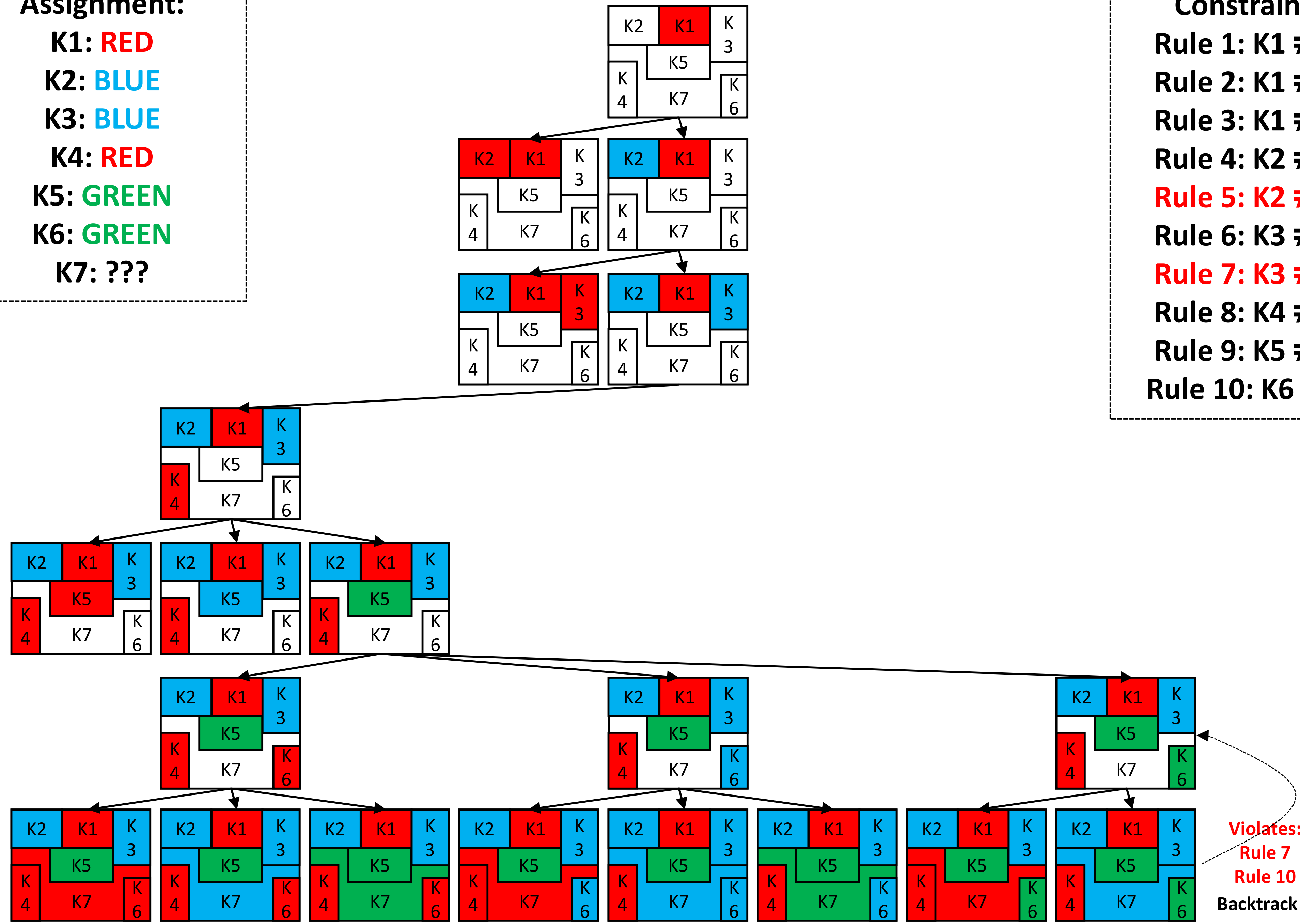


Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:

K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: GREEN
K7: ???



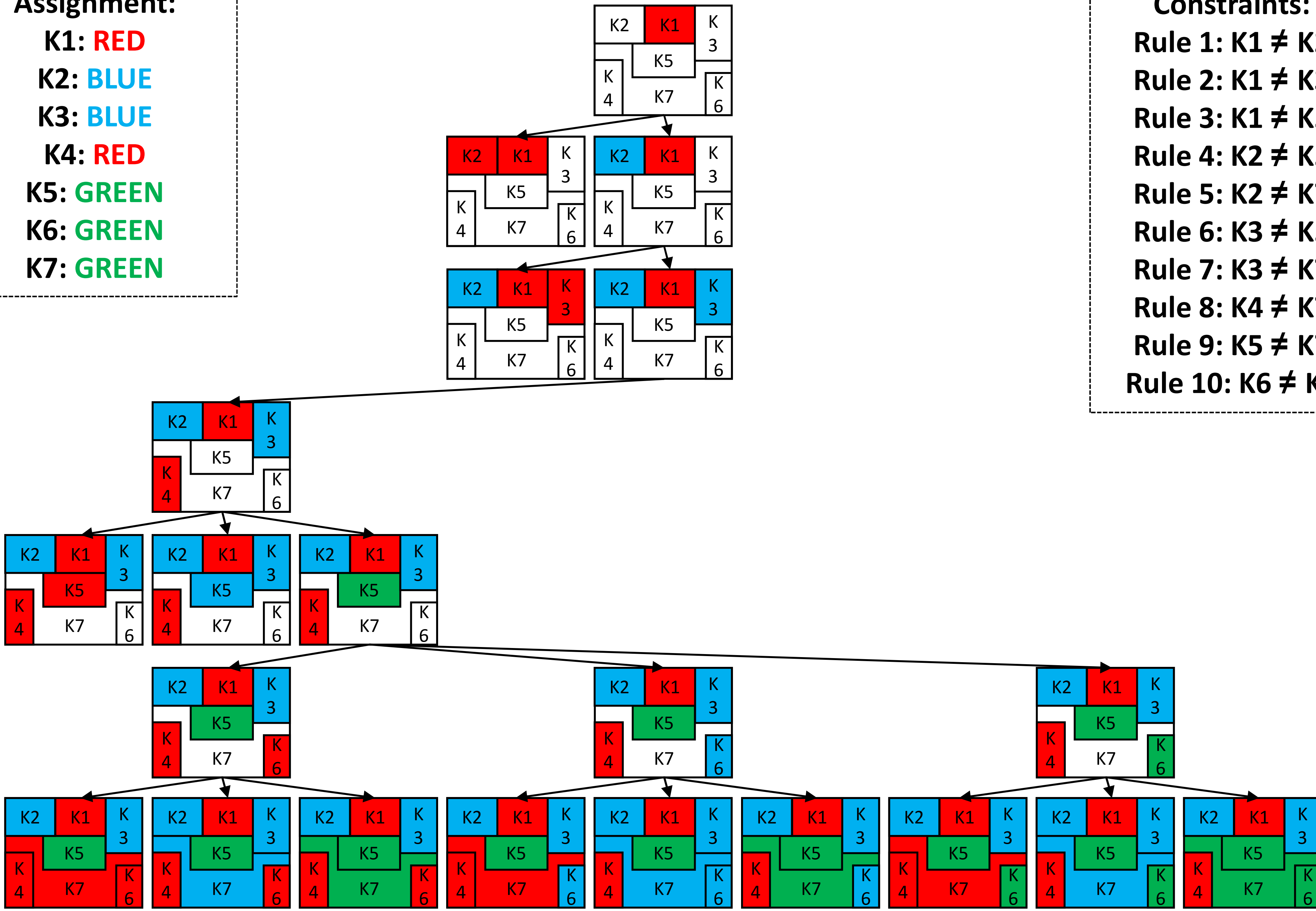
Constraints:

Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

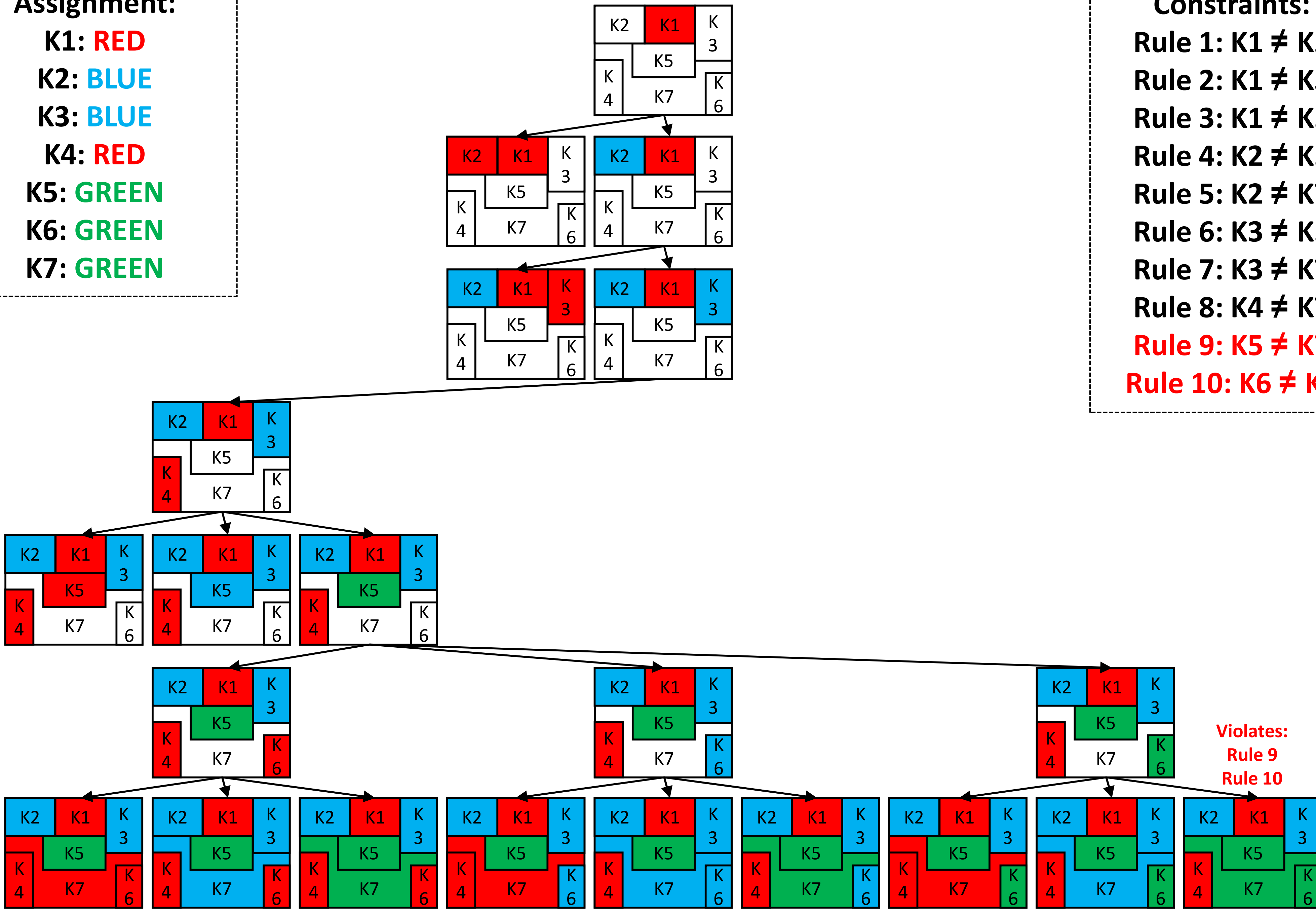
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: GREEN
K7: GREEN

Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order:** RED, BLUE, GREEN

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: GREEN
K7: GREEN

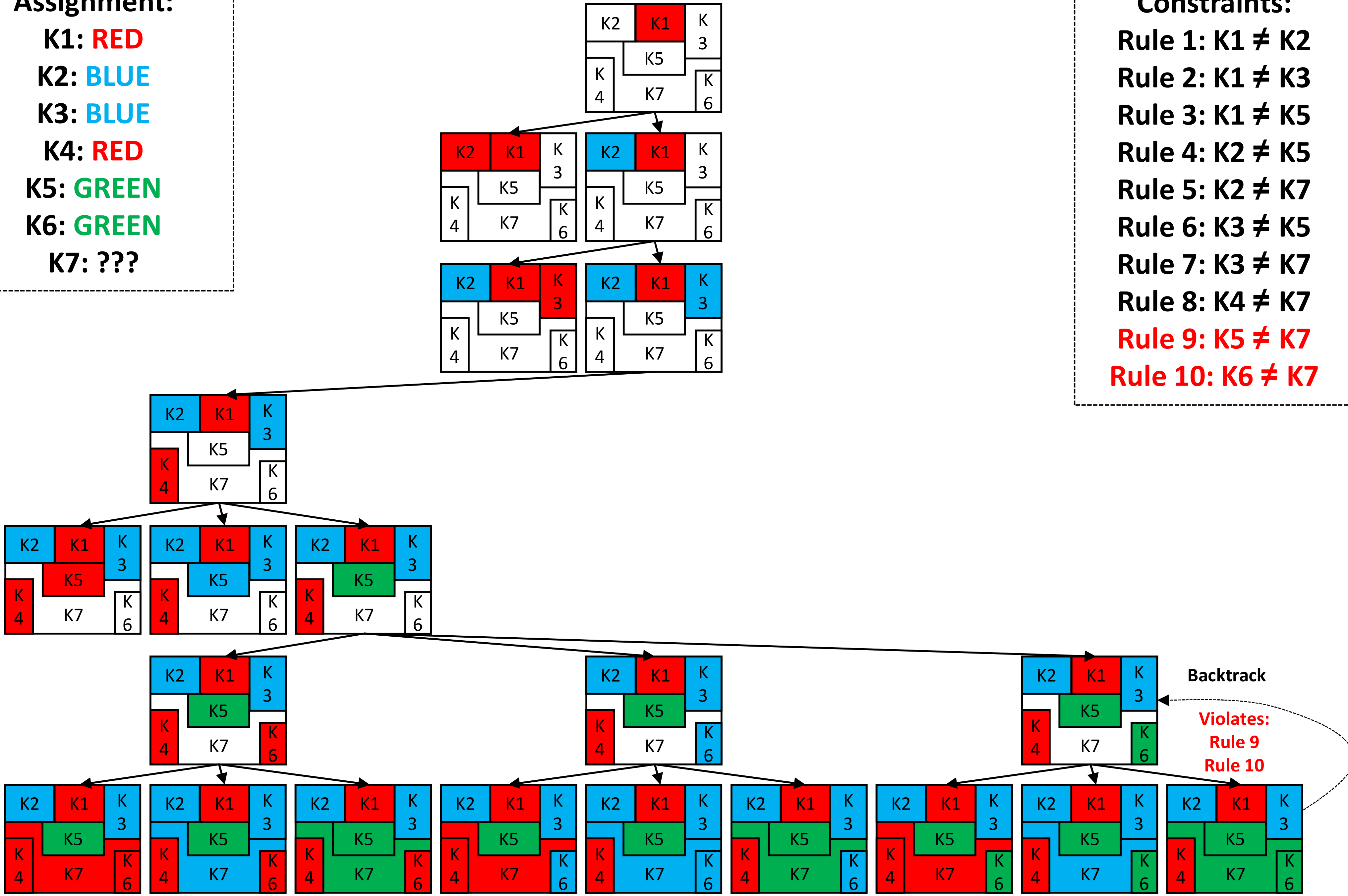


Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: GREEN
K7: ???

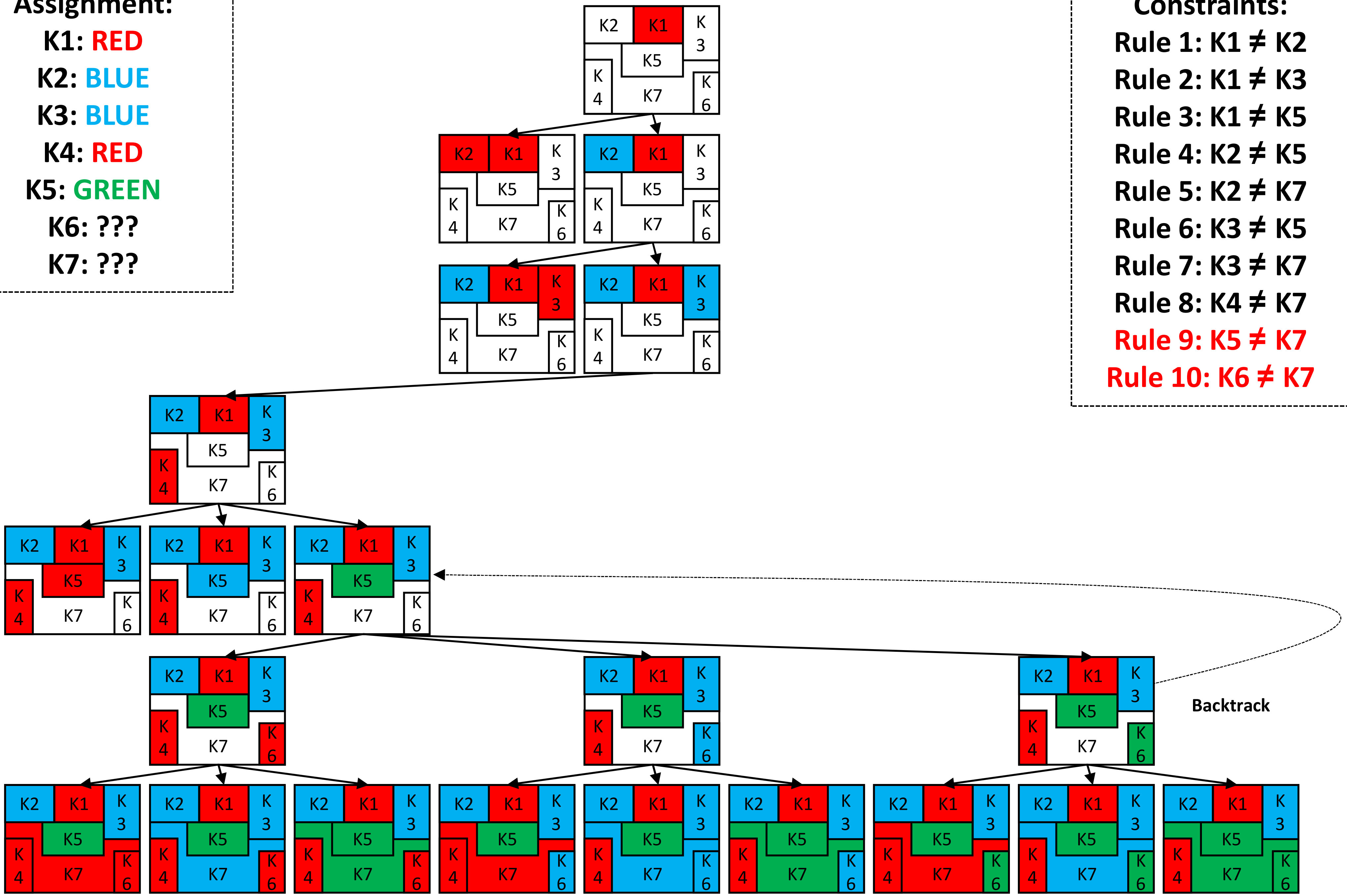
Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order:** RED, BLUE, GREEN

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: ???
K7: ???

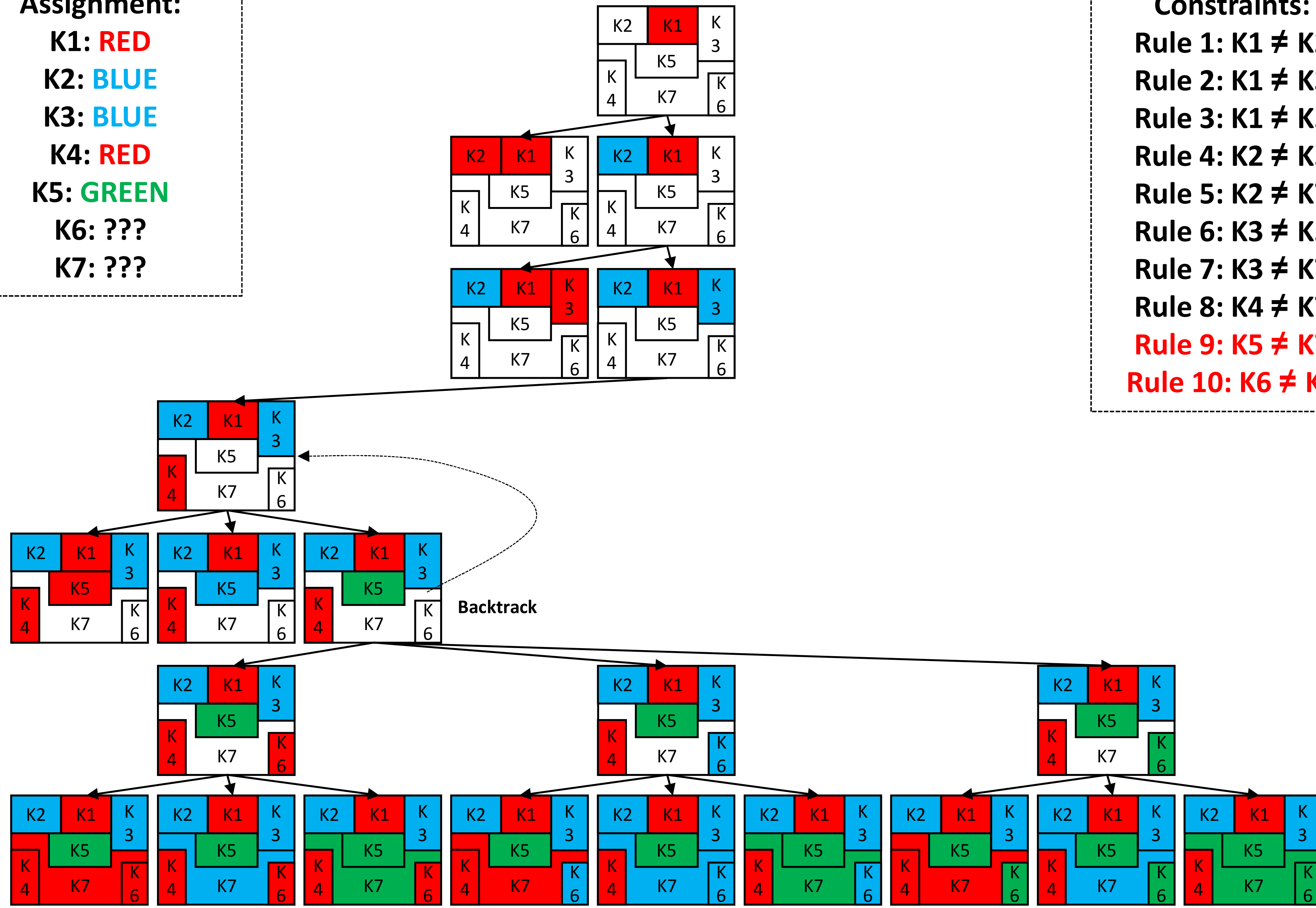
Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: RED
K5: GREEN
K6: ???
K7: ???

Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:

K1: RED

K2: BLUE

K3: BLUE

K4: RED

K5: ???

K6: ???

K7: ???

Constraints:

Rule 1: K1 ≠ K2

Rule 2: K1 ≠ K3

Rule 3: K1 ≠ K5

Rule 4: K2 ≠ K5

Rule 5: K2 ≠ K7

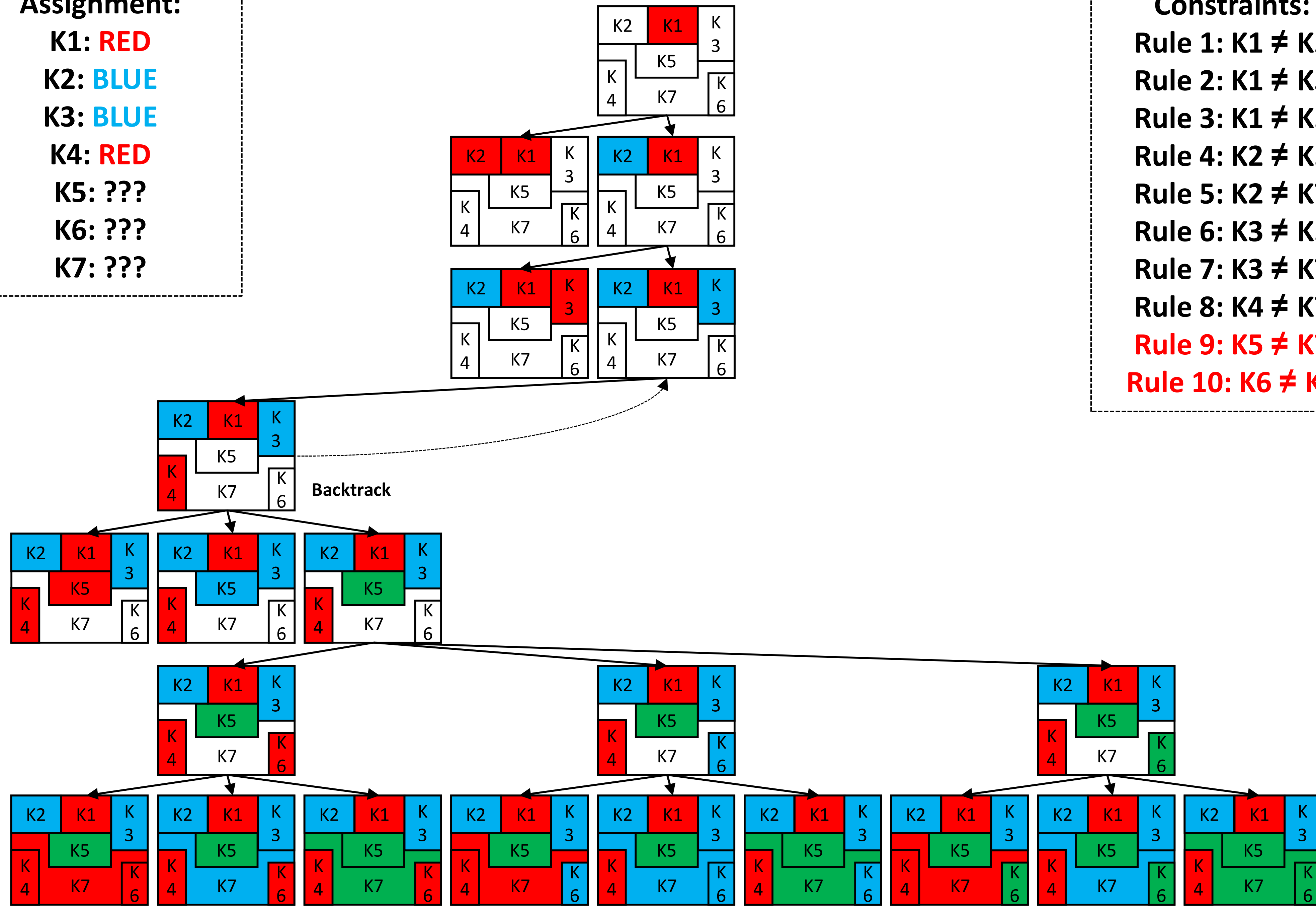
Rule 6: K3 ≠ K5

Rule 7: K3 ≠ K7

Rule 8: K4 ≠ K7

Rule 9: K5 ≠ K7

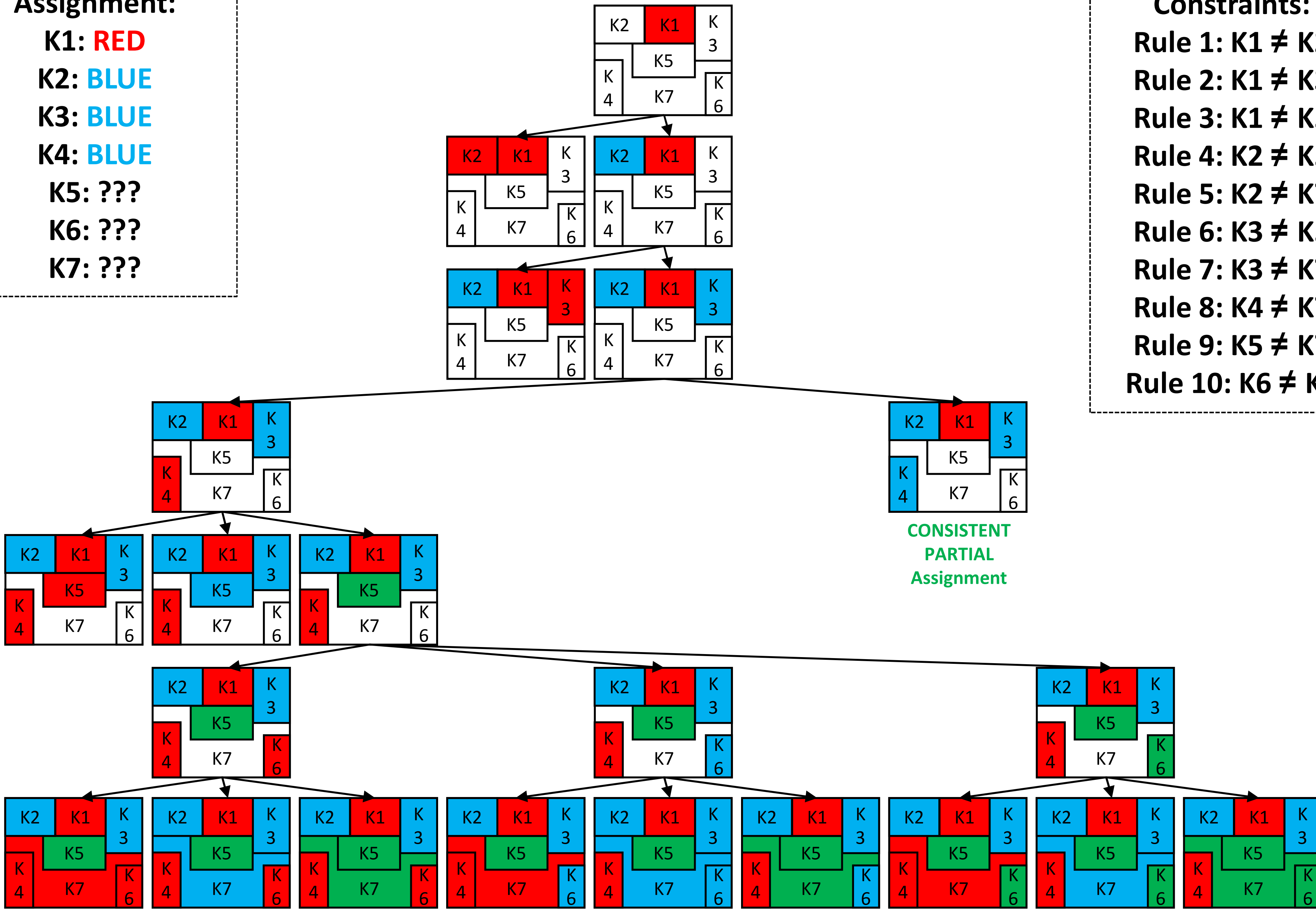
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order:** RED, BLUE, GREEN

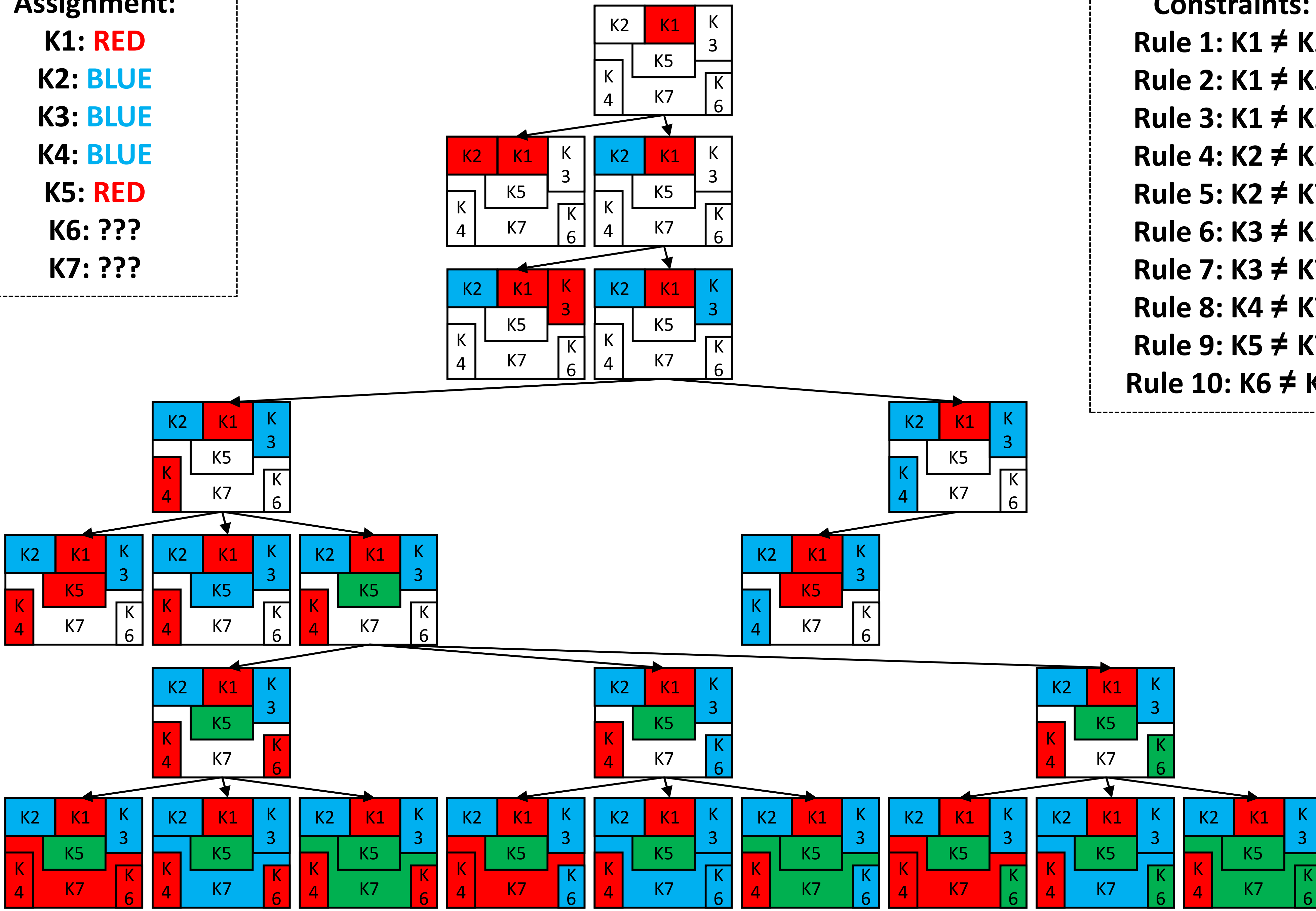
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: ???
K6: ???
K7: ???

Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: RED
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:

K1: RED

K2: BLUE

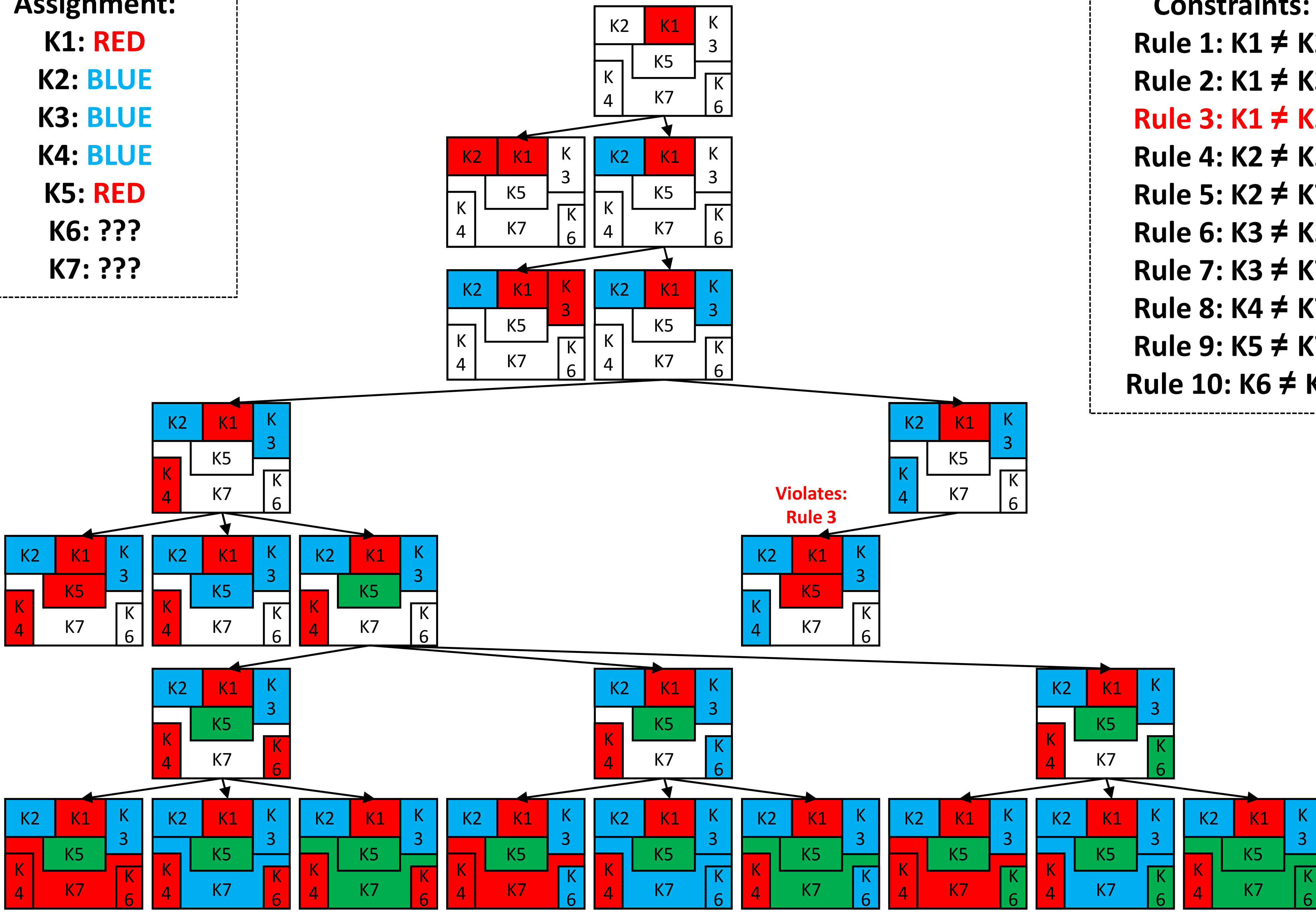
K3: BLUE

K4: BLUE

K5: RED

K6: ???

K7: ???



Constraints:

Rule 1: K1 ≠ K2

Rule 2: K1 ≠ K3

Rule 3: K1 ≠ K5

Rule 4: K2 ≠ K5

Rule 5: K2 ≠ K7

Rule 6: K3 ≠ K5

Rule 7: K3 ≠ K7

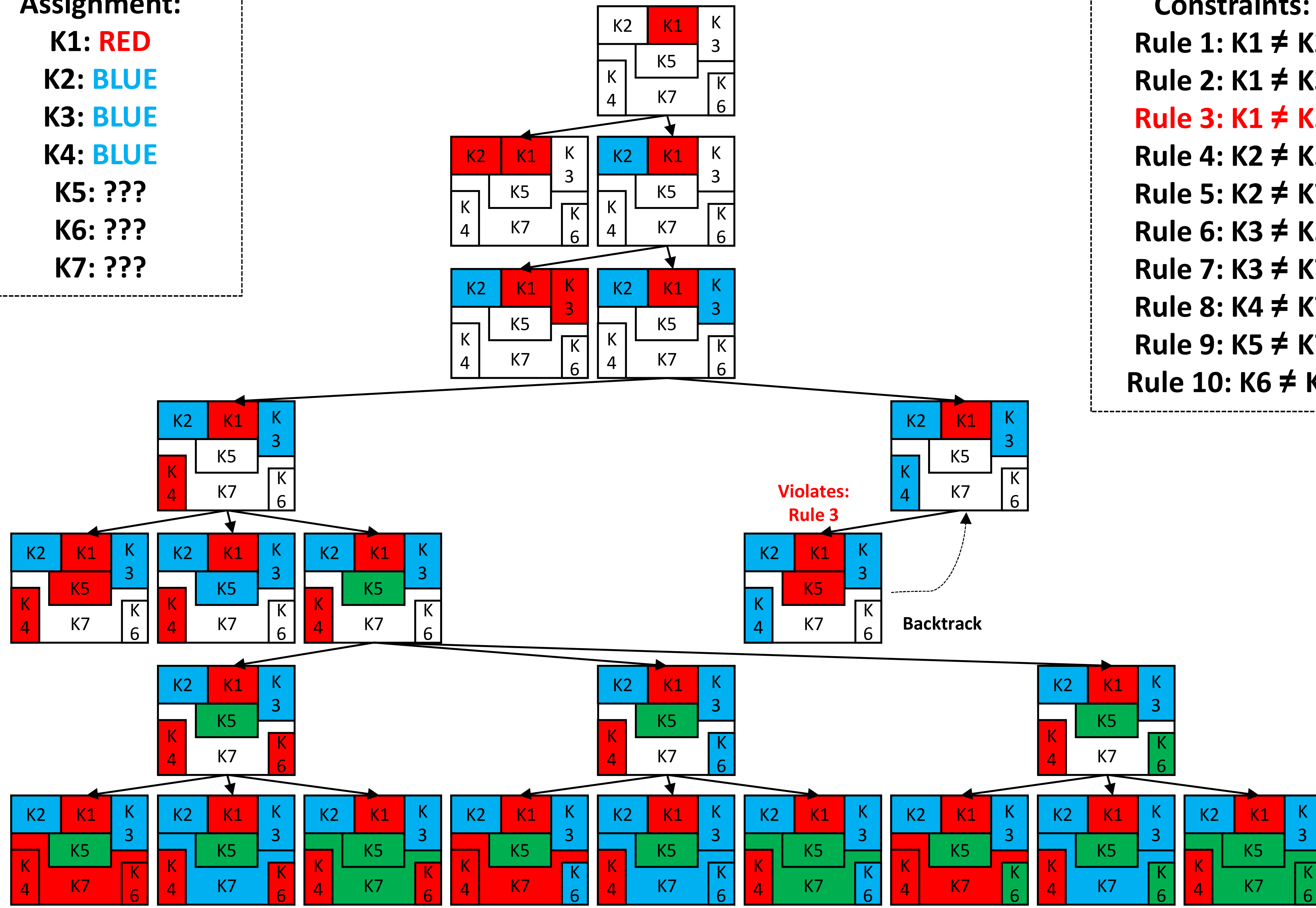
Rule 8: K4 ≠ K7

Rule 9: K5 ≠ K7

Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

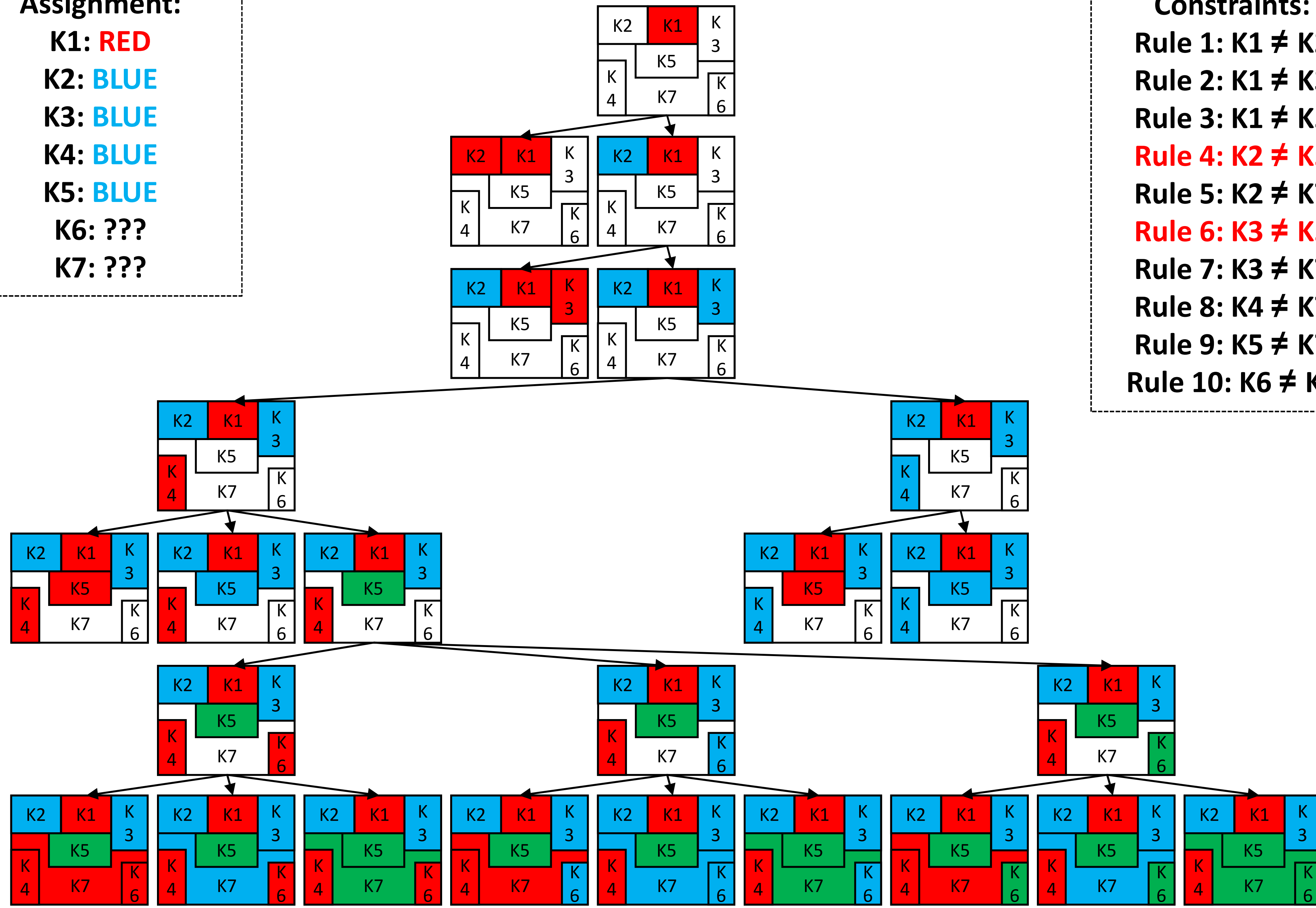
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: ???
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: BLUE
K6: ???
K7: ???

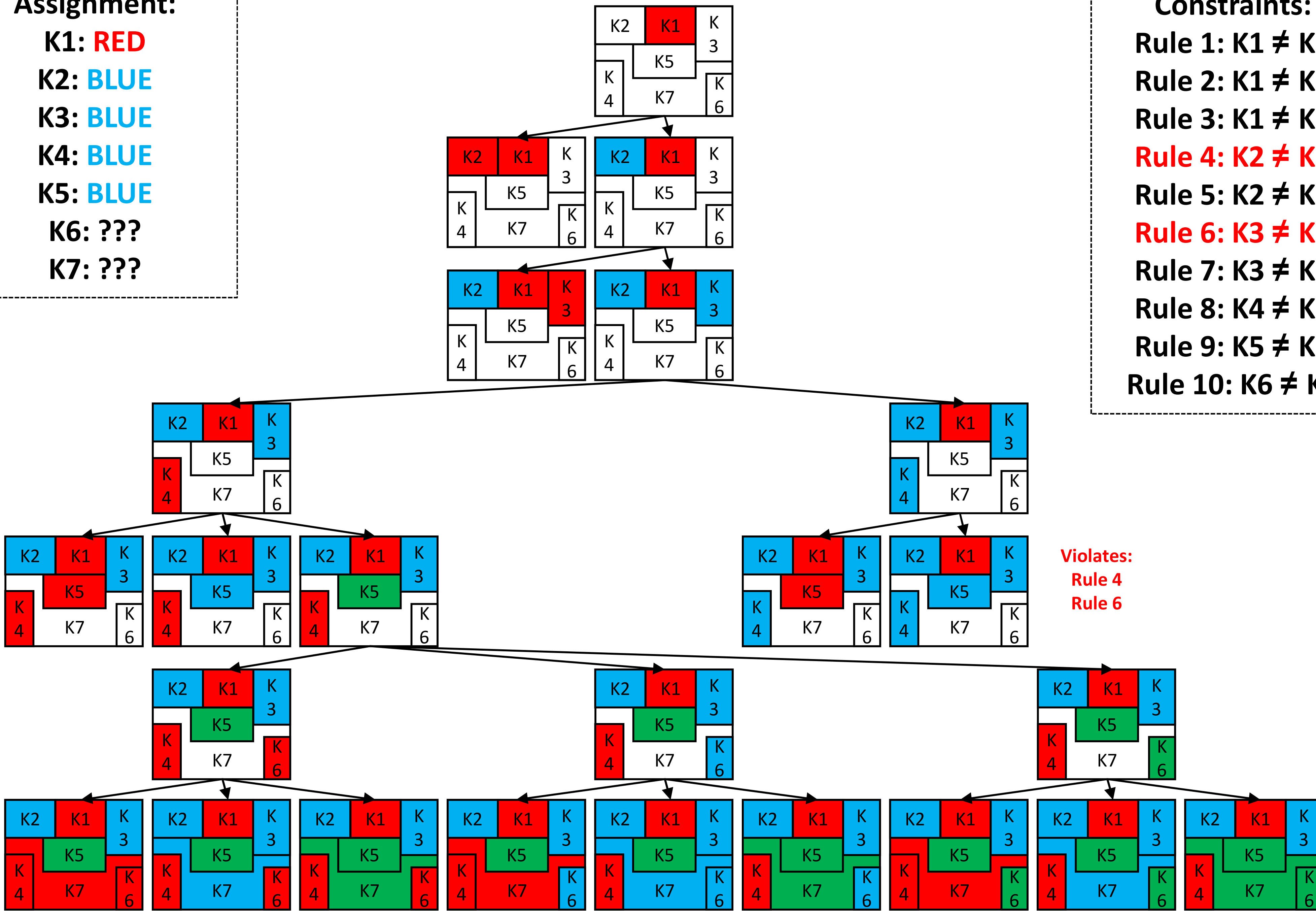


Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

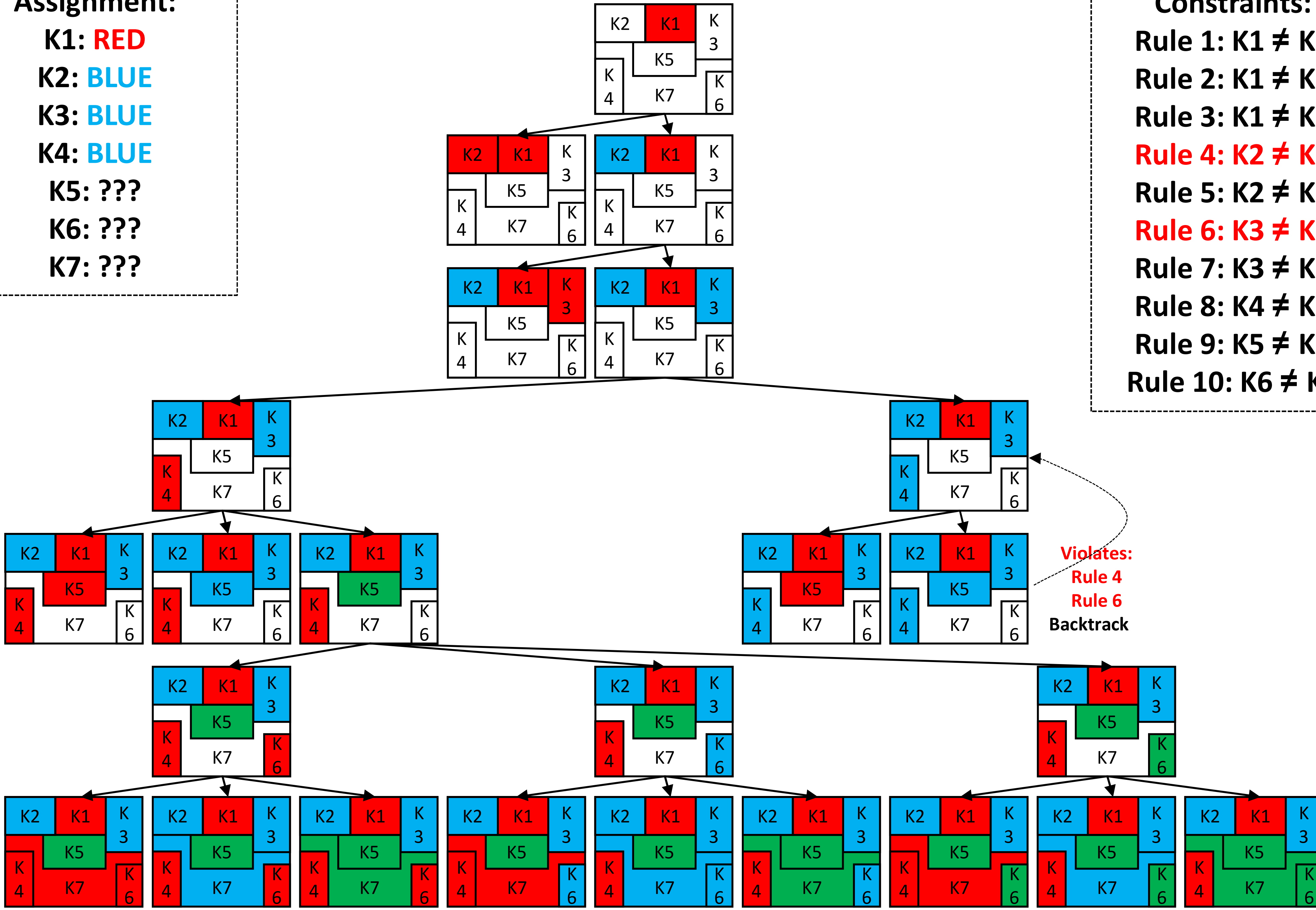
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: BLUE
K6: ???
K7: ???

Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: ???
K6: ???
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:

K1: RED

K2: BLUE

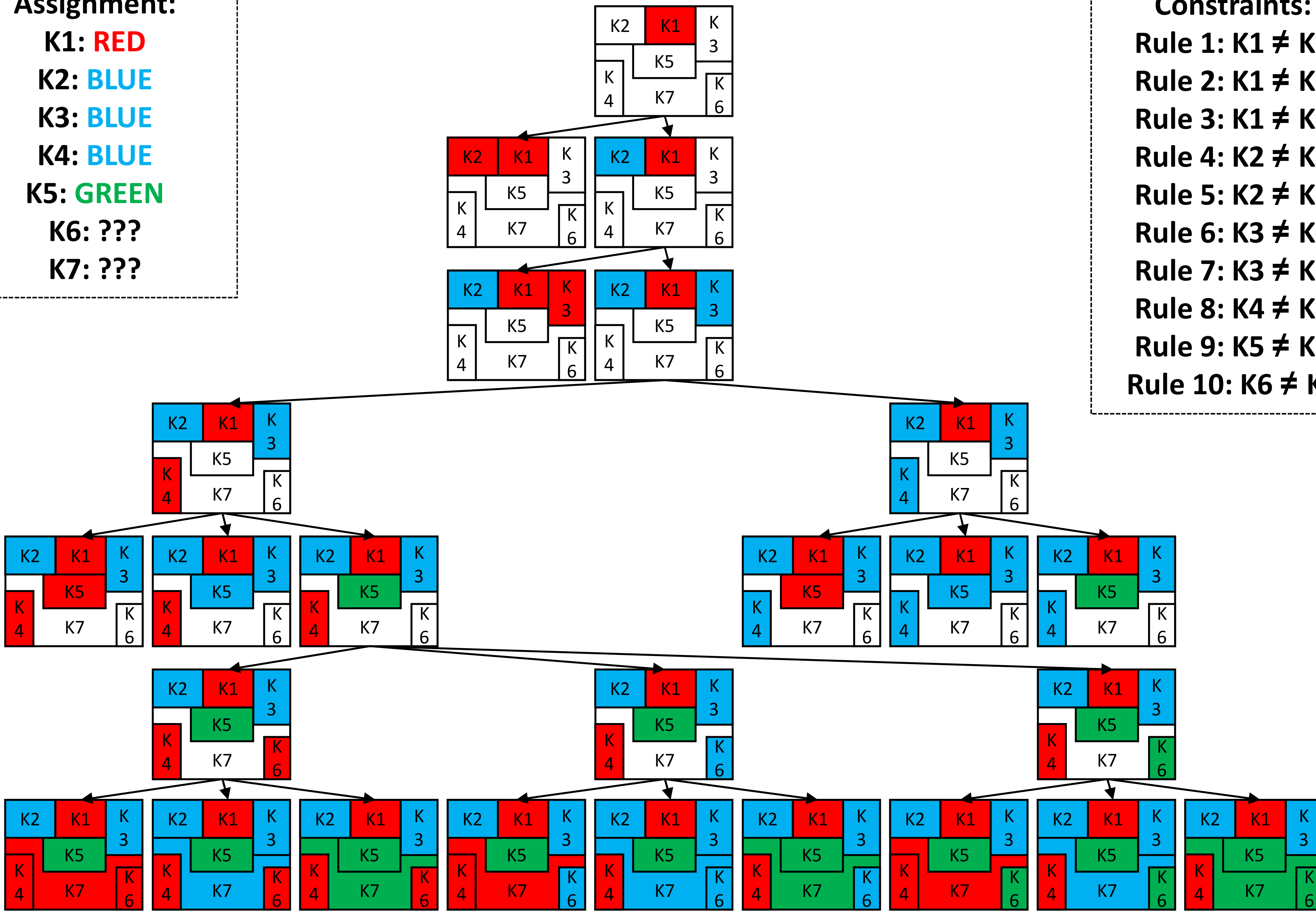
K3: BLUE

K4: BLUE

K5: GREEN

K6: ???

K7: ???



Constraints:

Rule 1: K1 ≠ K2

Rule 2: K1 ≠ K3

Rule 3: K1 ≠ K5

Rule 4: K2 ≠ K5

Rule 5: K2 ≠ K7

Rule 6: K3 ≠ K5

Rule 7: K3 ≠ K7

Rule 8: K4 ≠ K7

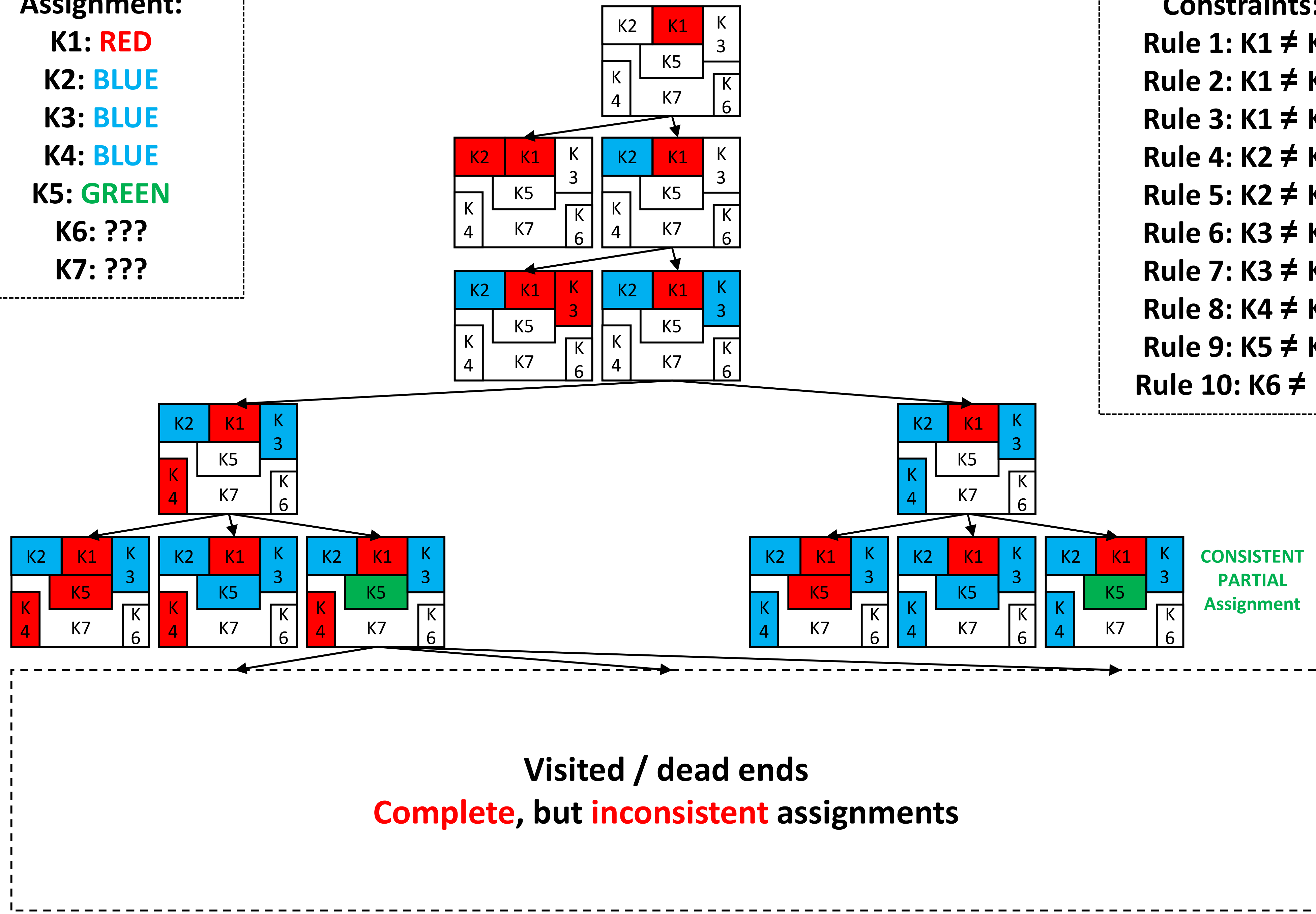
Rule 9: K5 ≠ K7

Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

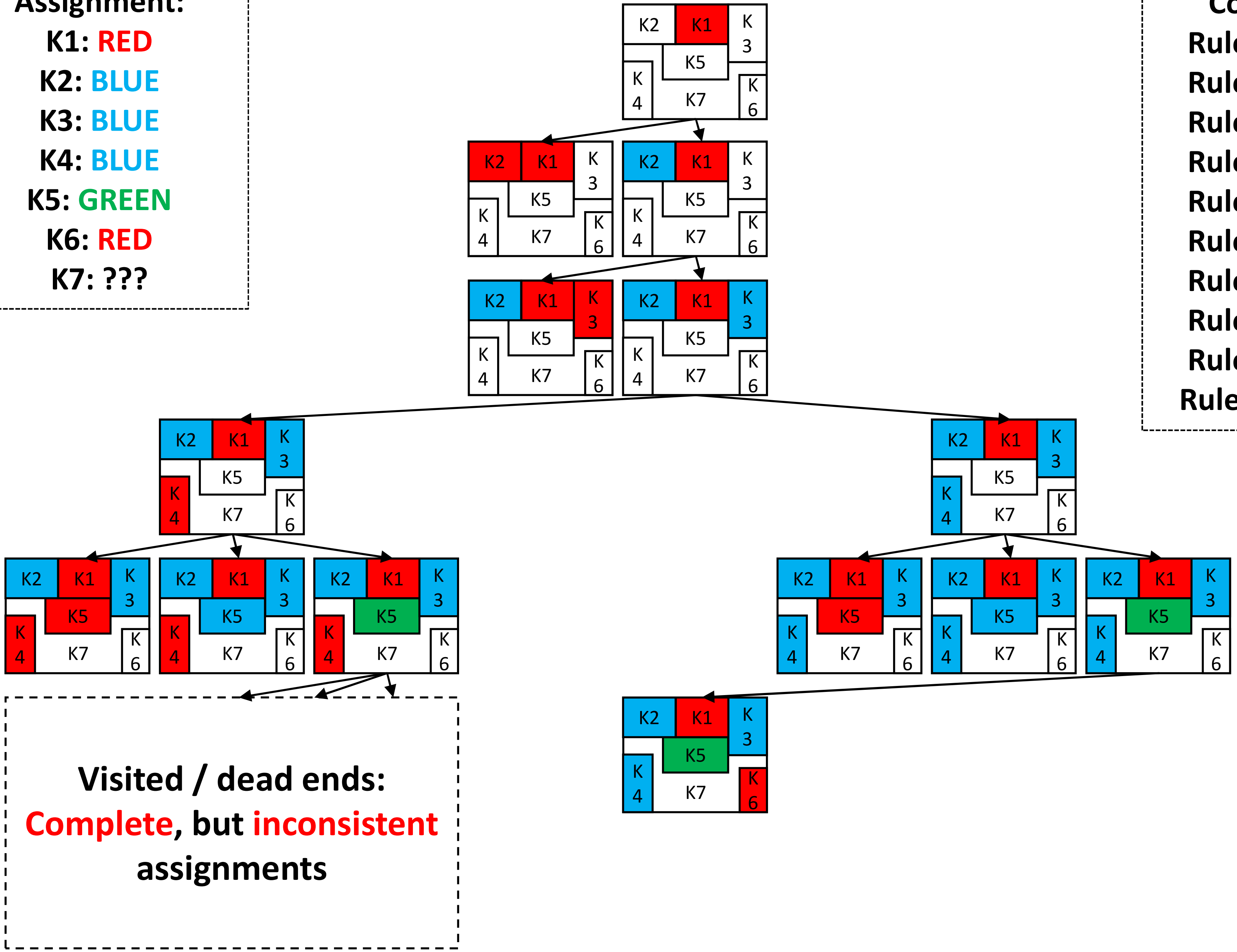
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: ???
K7: ???

Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order:** RED, BLUE, GREEN

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: RED
K7: ???

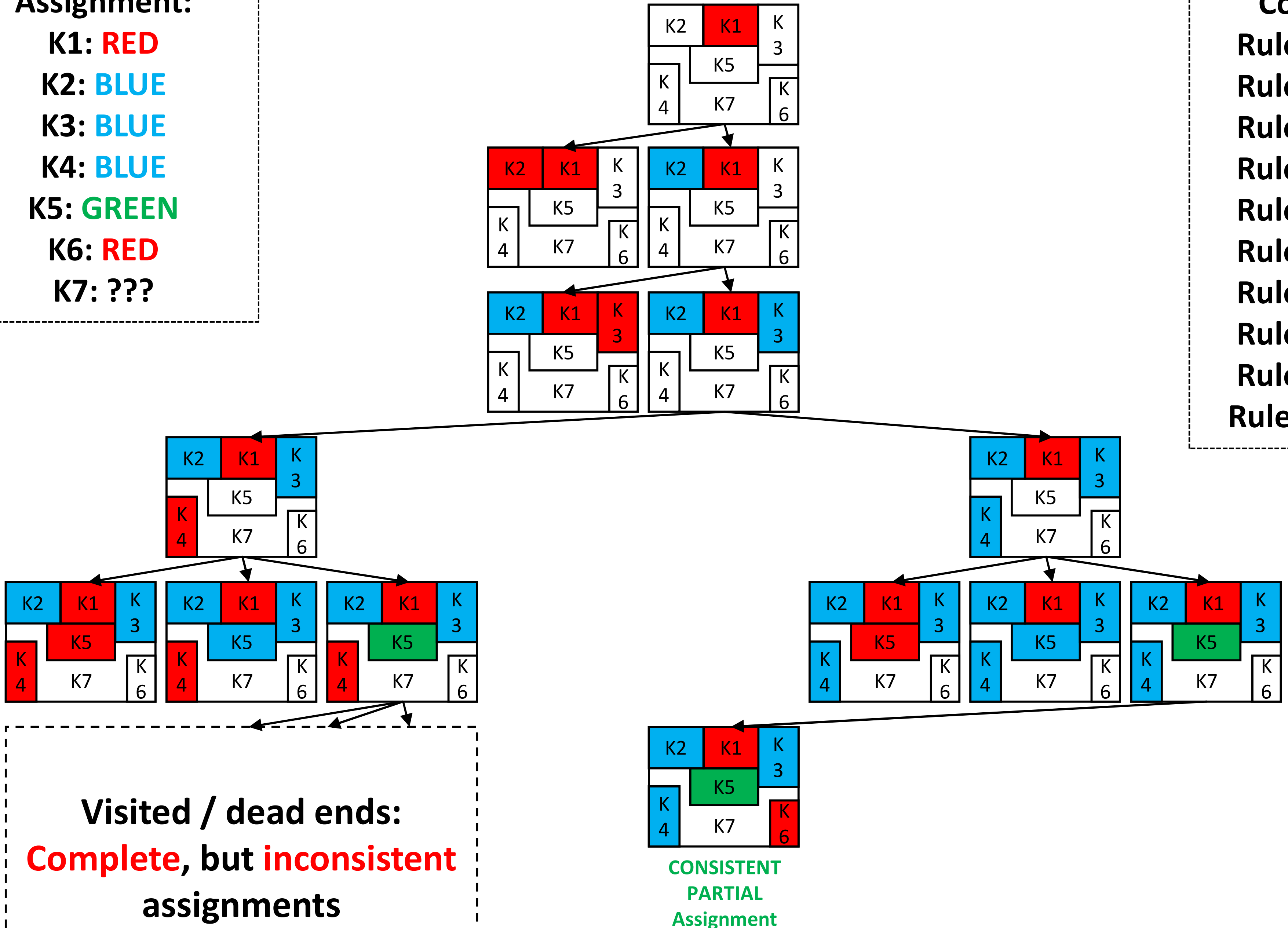


Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Visited / dead ends:
Complete, but inconsistent assignments

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: RED
K7: ???



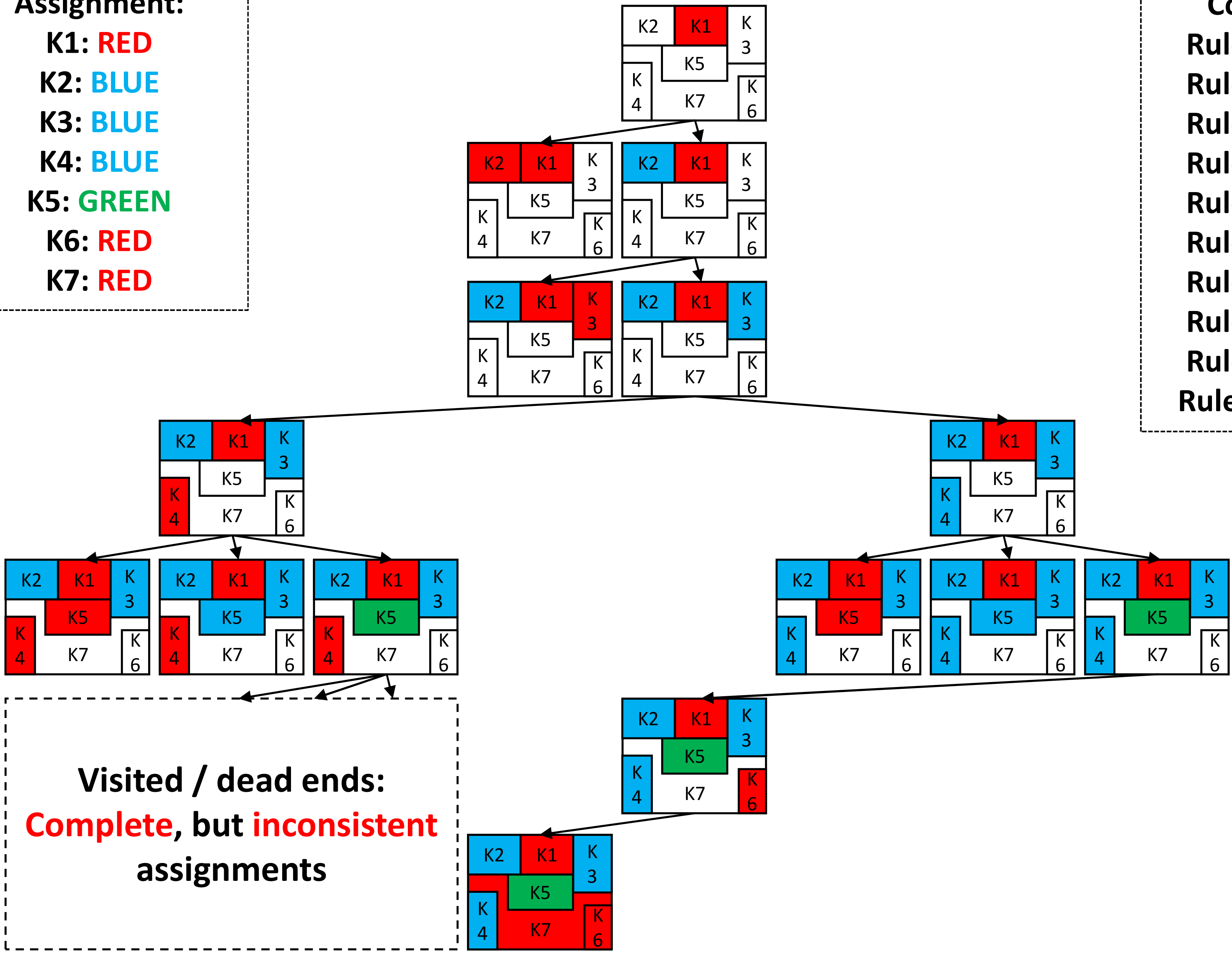
Visited / dead ends:
Complete, but inconsistent assignments

Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: RED
K7: RED

Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

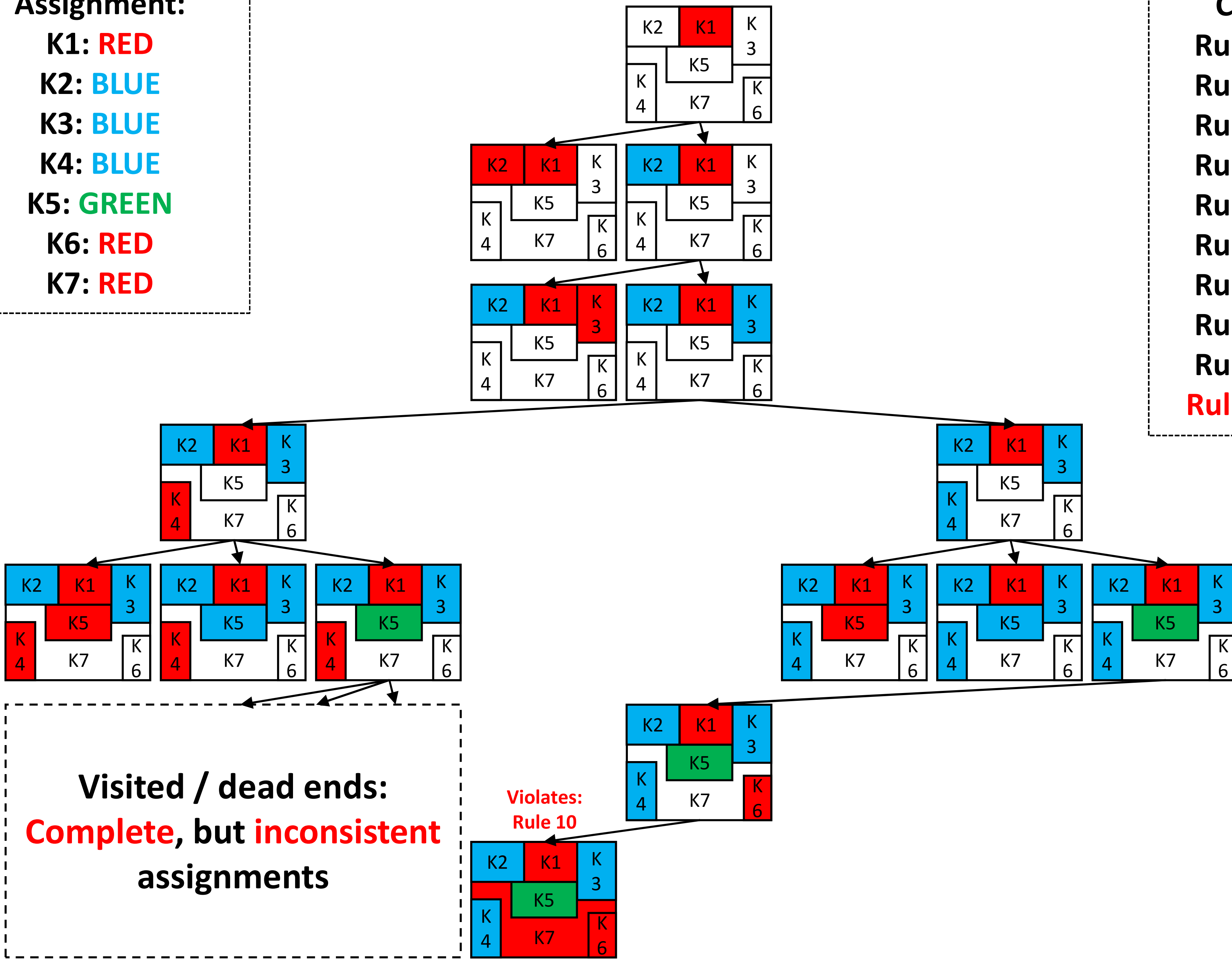


Visited / dead ends:
Complete, but inconsistent assignments

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

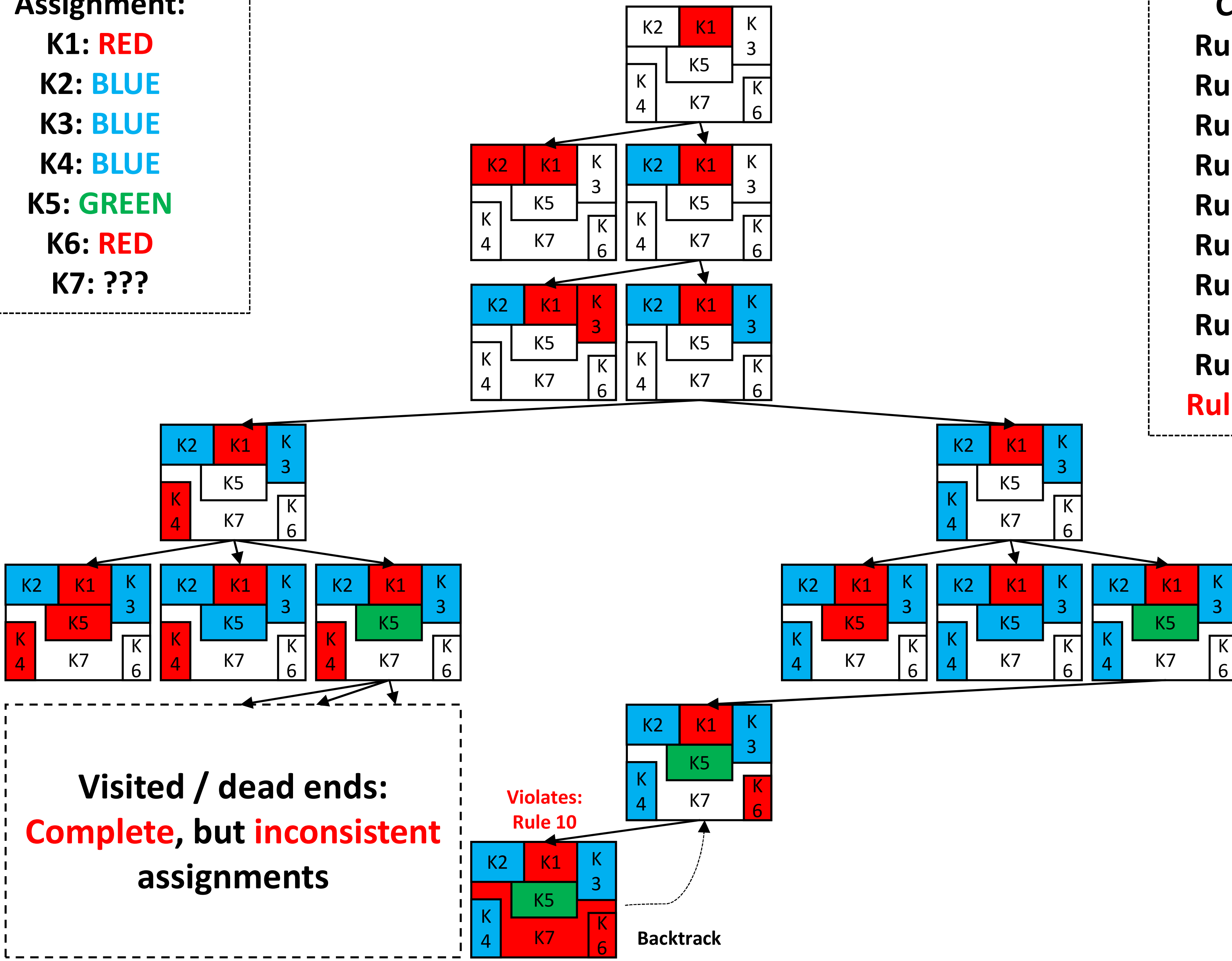
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: RED
K7: RED

Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order:** RED, BLUE, GREEN

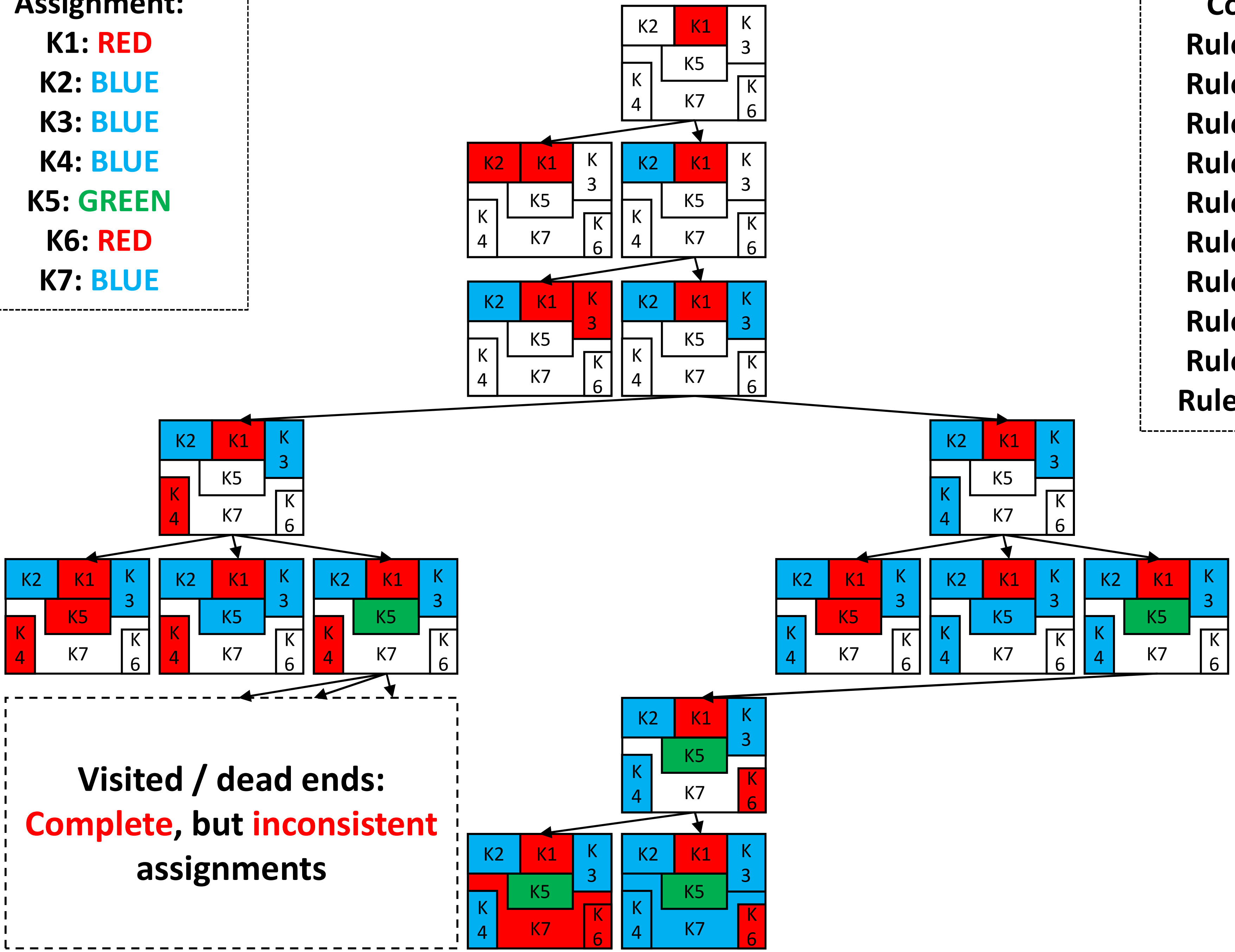
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: RED
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

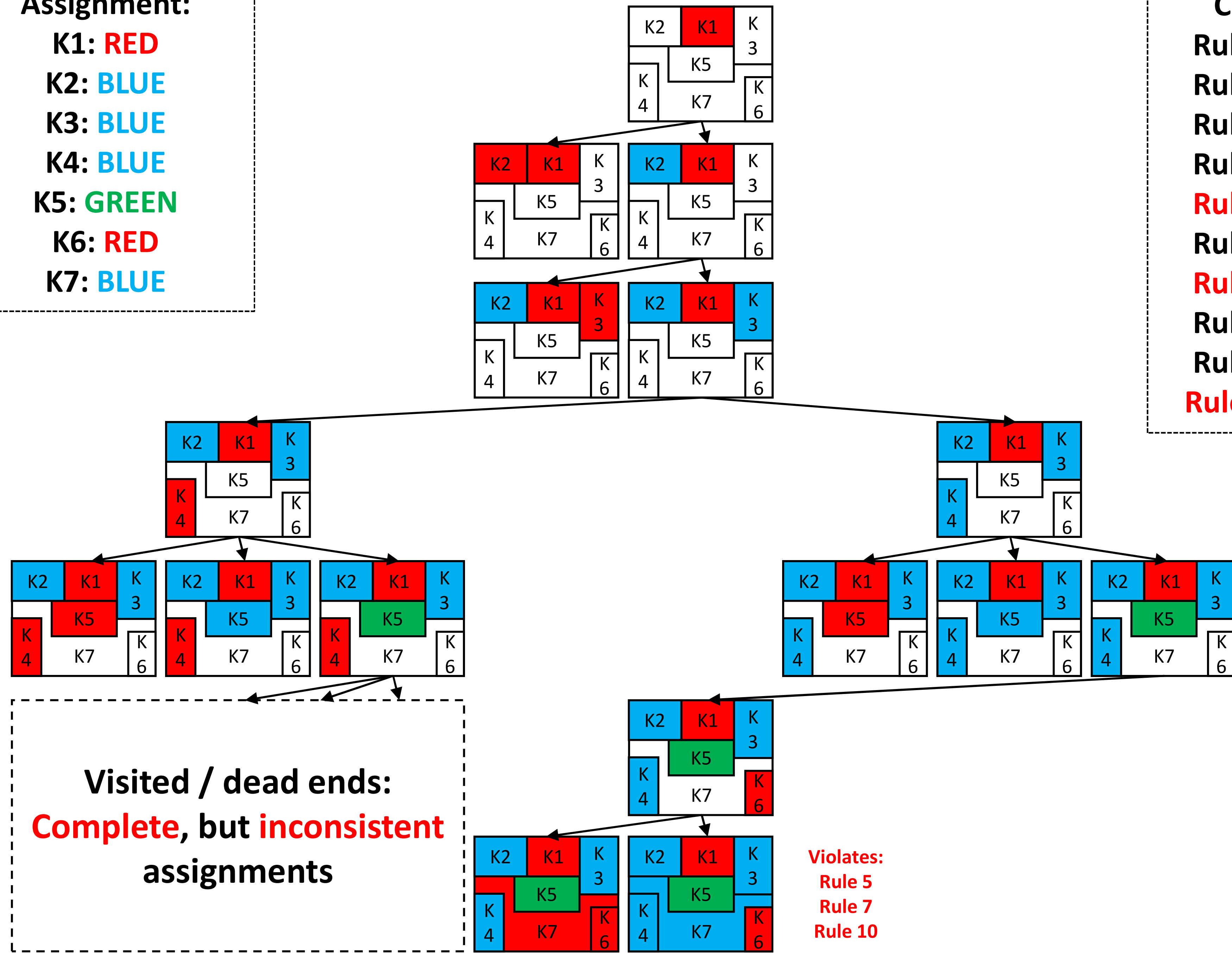
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: RED
K7: BLUE



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: RED
K7: BLUE

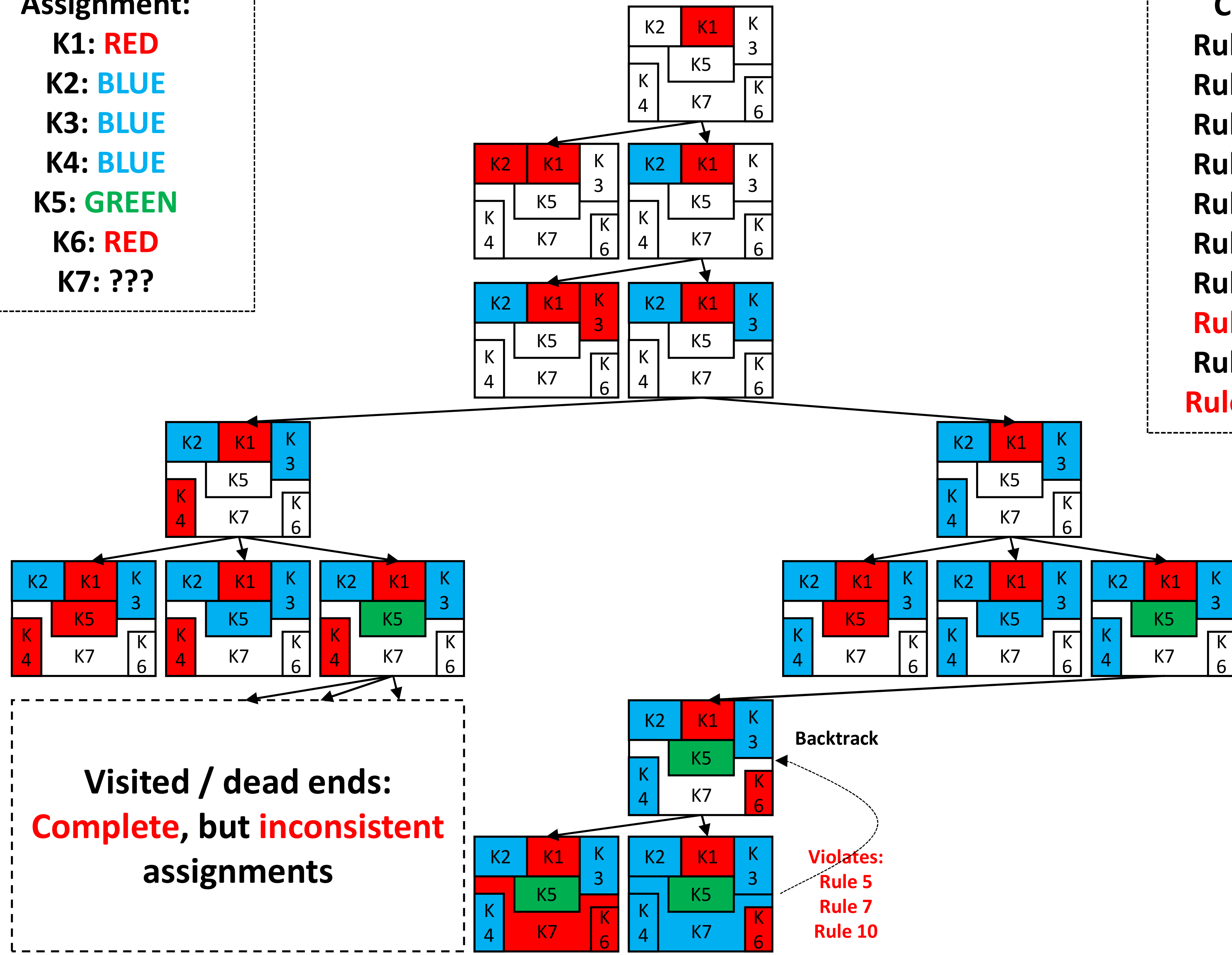


Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

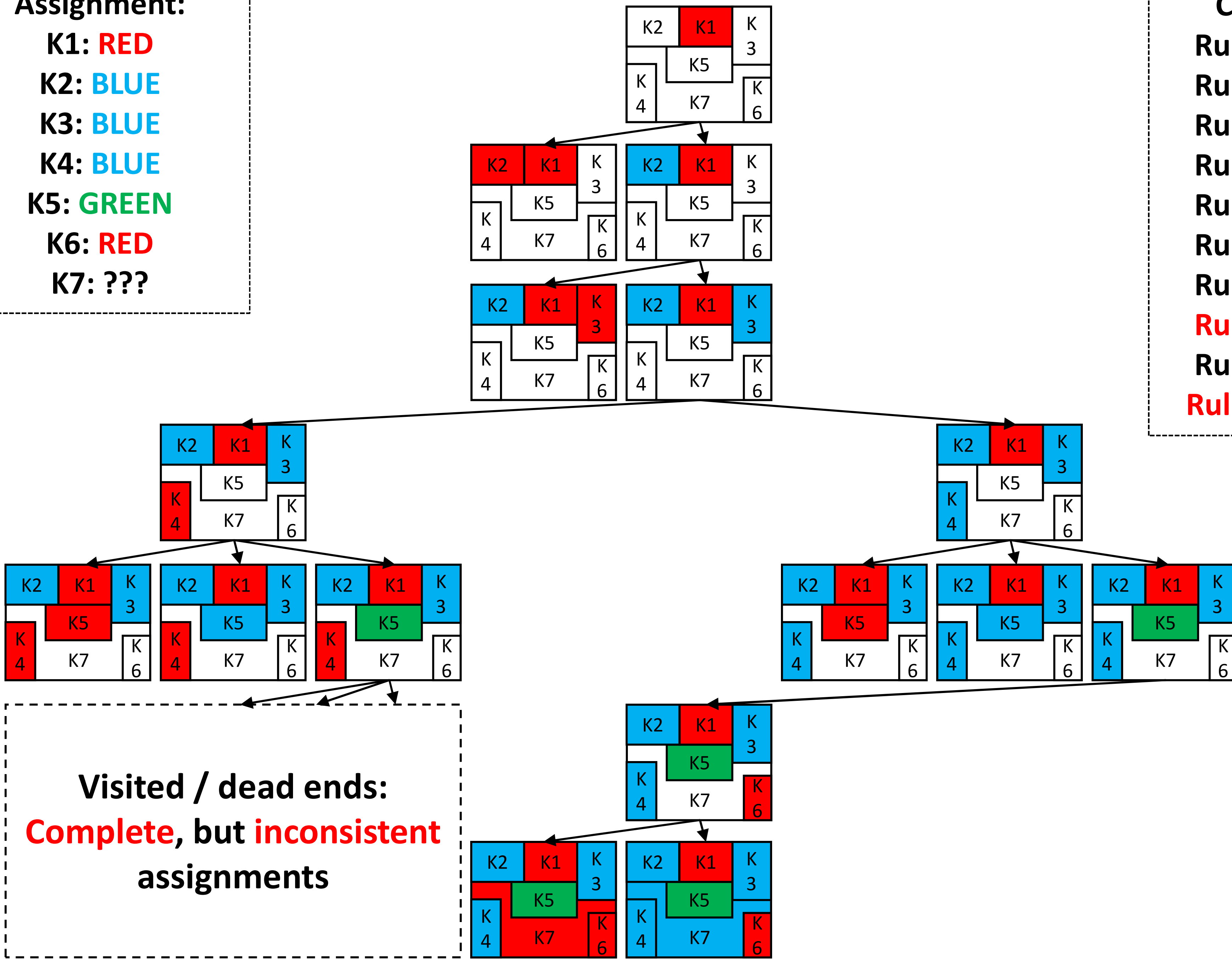
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: RED
K7: ???

Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order:** RED, BLUE, GREEN

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: RED
K7: ???



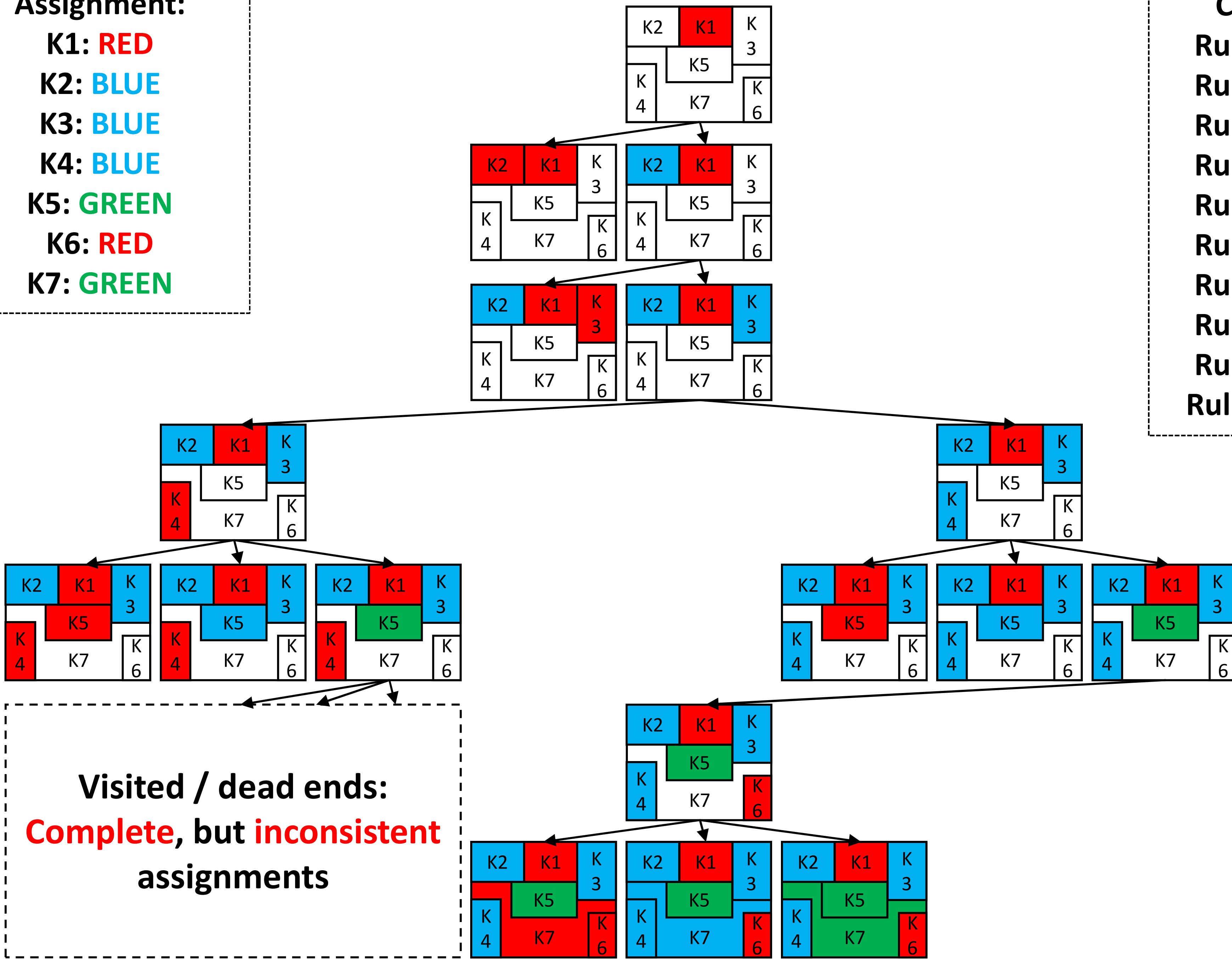
Visited / dead ends:
Complete, but inconsistent assignments

Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: RED
K7: GREEN

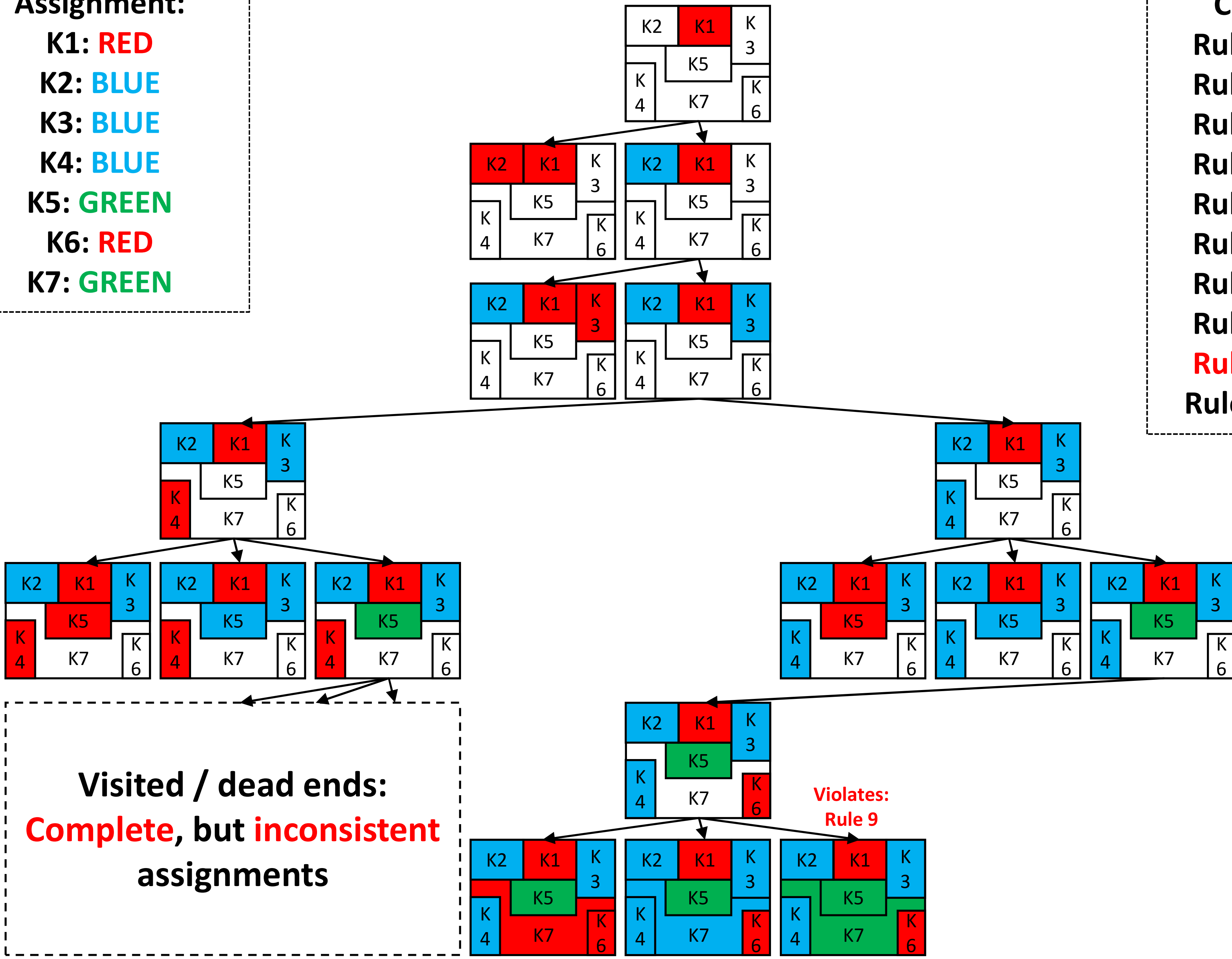
Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:

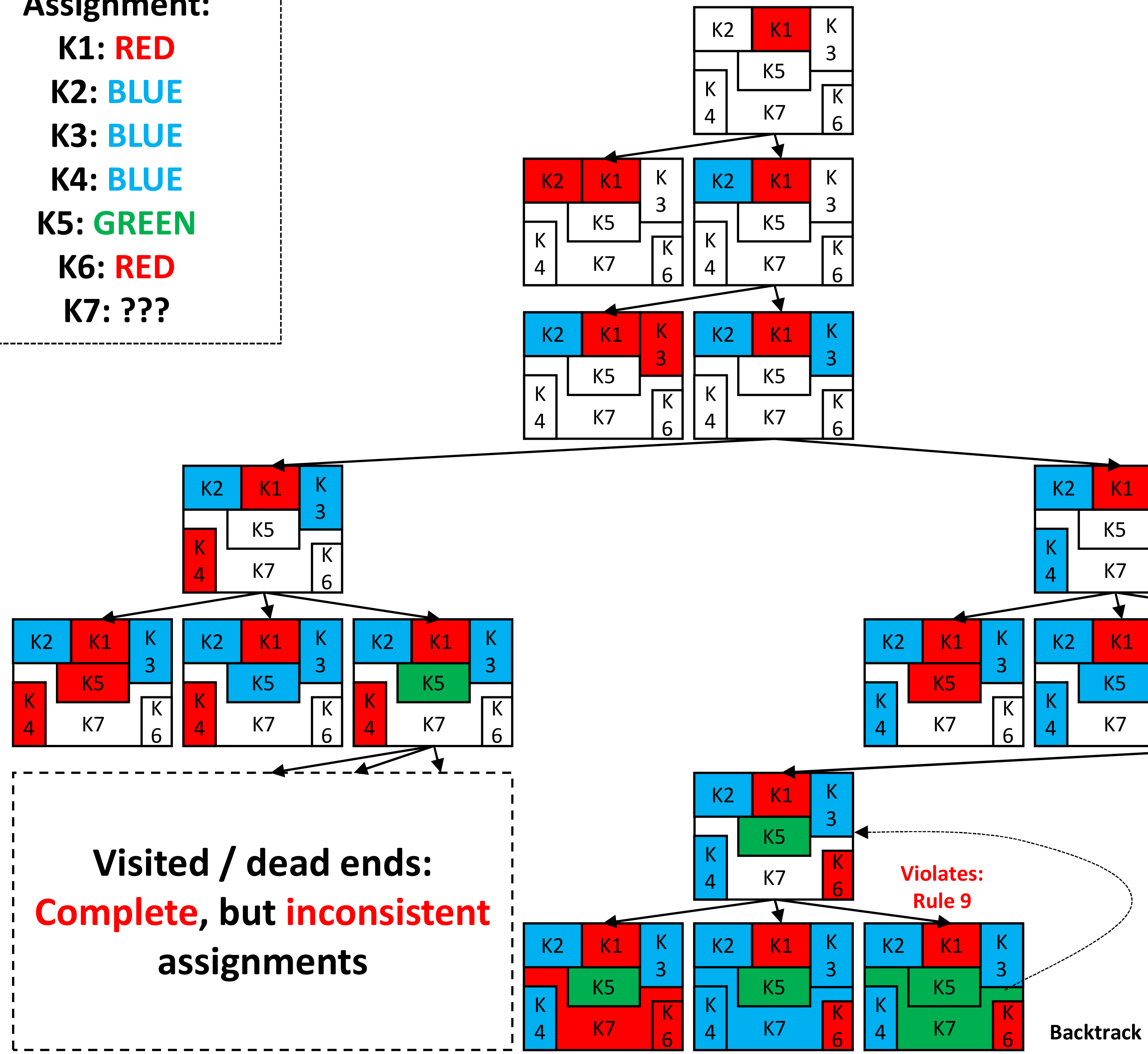
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: RED
K7: GREEN



Constraints:

Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: RED
K7: ???

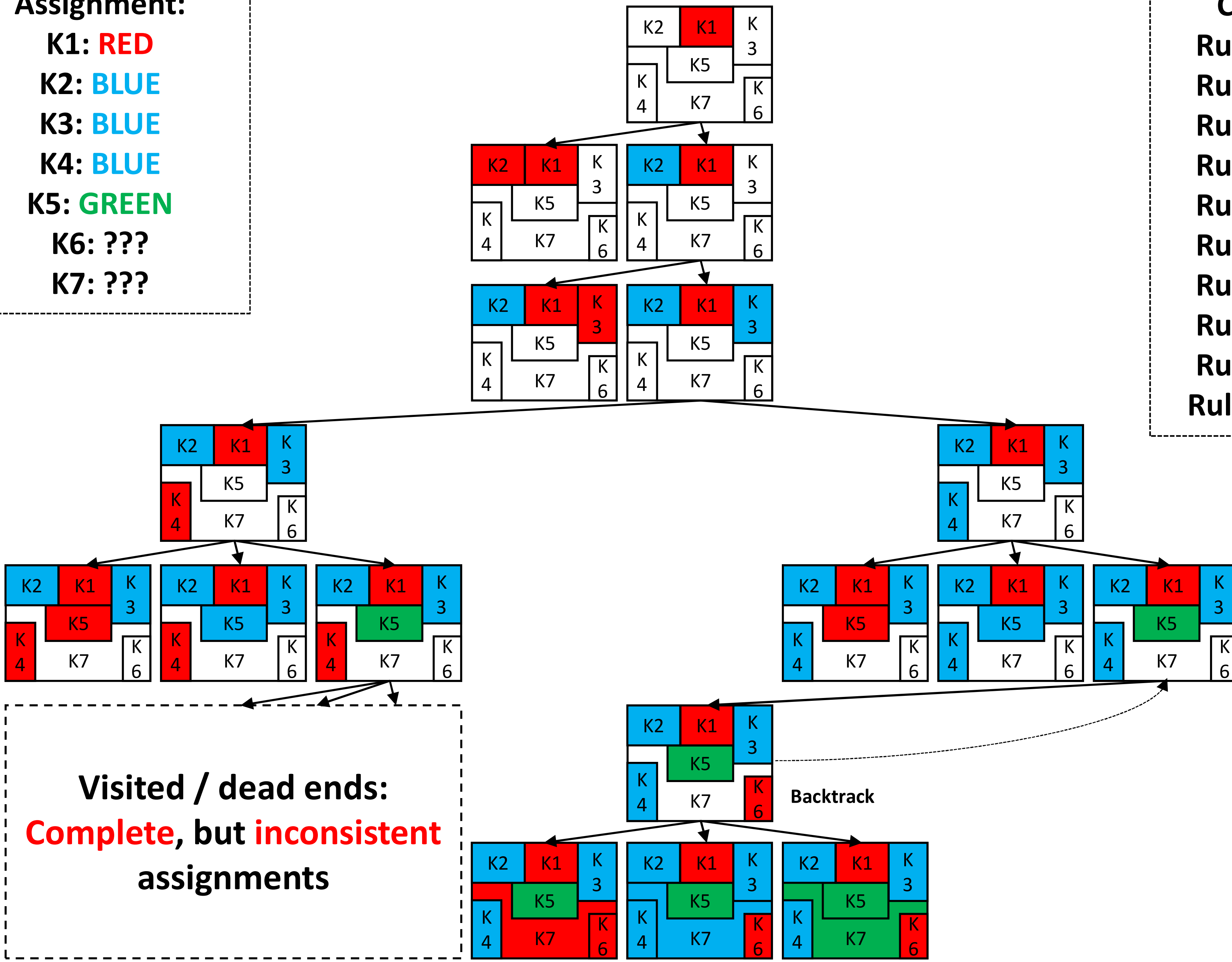


Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

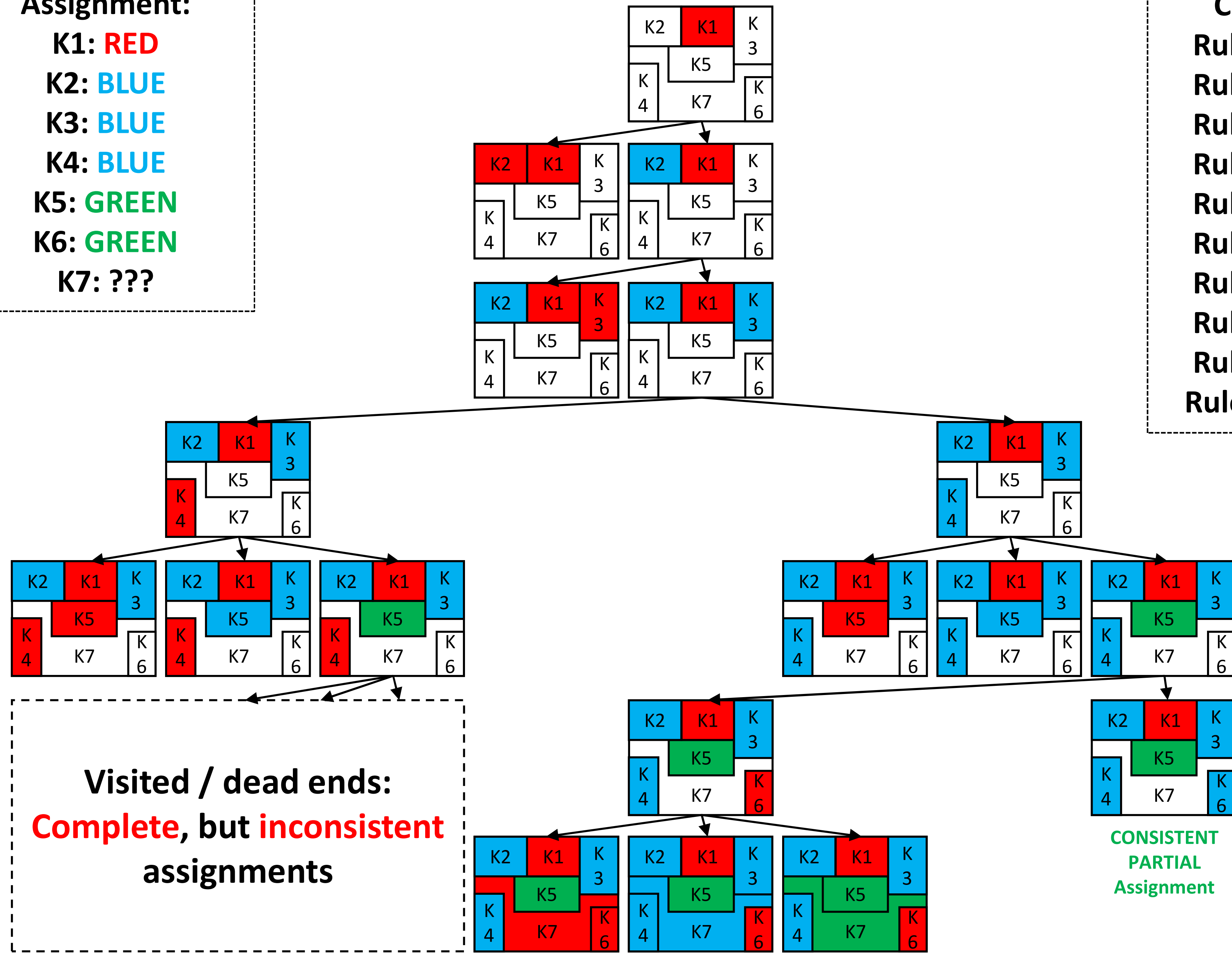
Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: ???
K7: ???

Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order:** RED, BLUE, GREEN

Assignment:
K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: GREEN
K7: ???



Constraints:
Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7

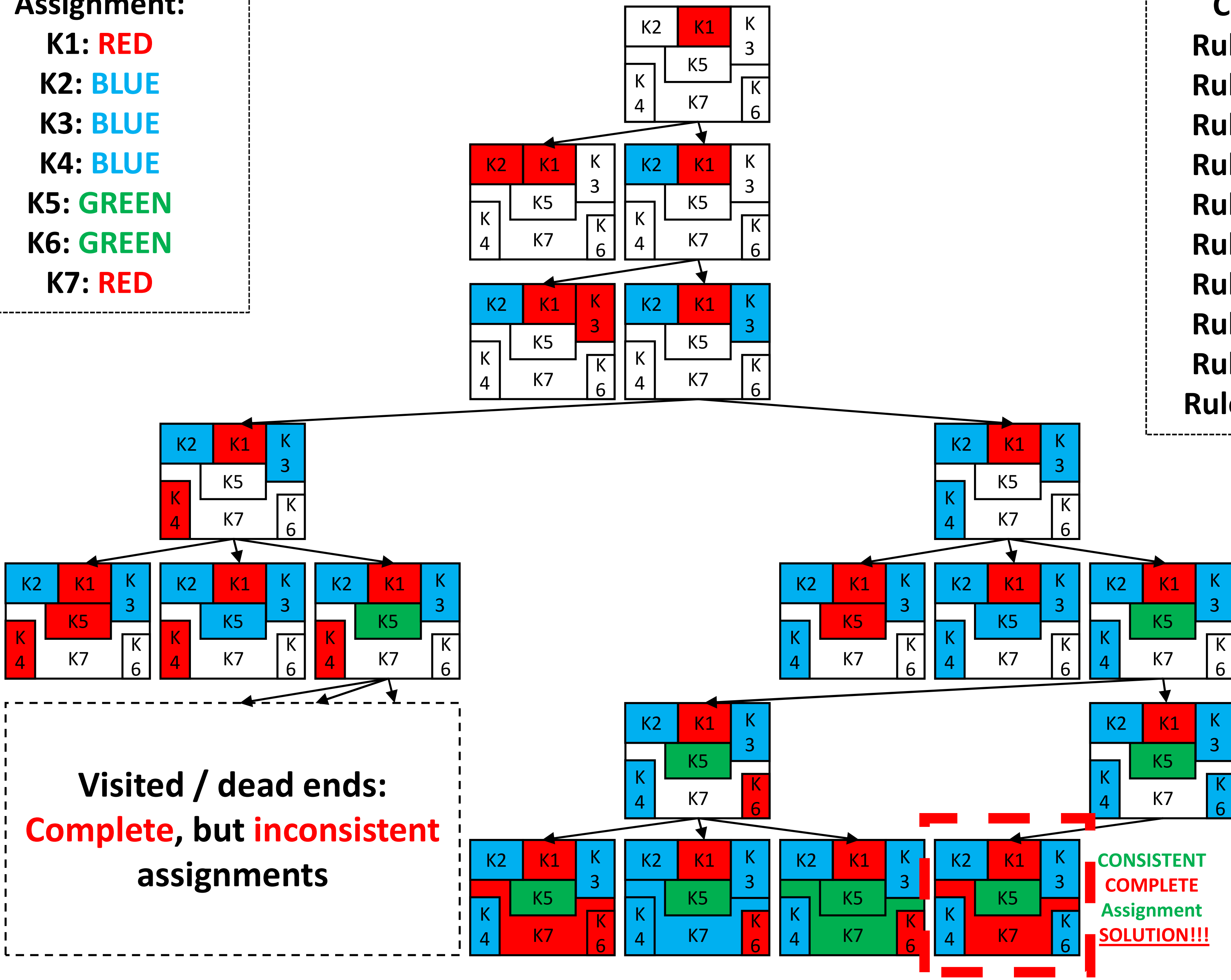
Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Assignment:

K1: RED
K2: BLUE
K3: BLUE
K4: BLUE
K5: GREEN
K6: GREEN
K7: RED

Constraints:

Rule 1: K1 ≠ K2
Rule 2: K1 ≠ K3
Rule 3: K1 ≠ K5
Rule 4: K2 ≠ K5
Rule 5: K2 ≠ K7
Rule 6: K3 ≠ K5
Rule 7: K3 ≠ K7
Rule 8: K4 ≠ K7
Rule 9: K5 ≠ K7
Rule 10: K6 ≠ K7



Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**

Can We Do Better?

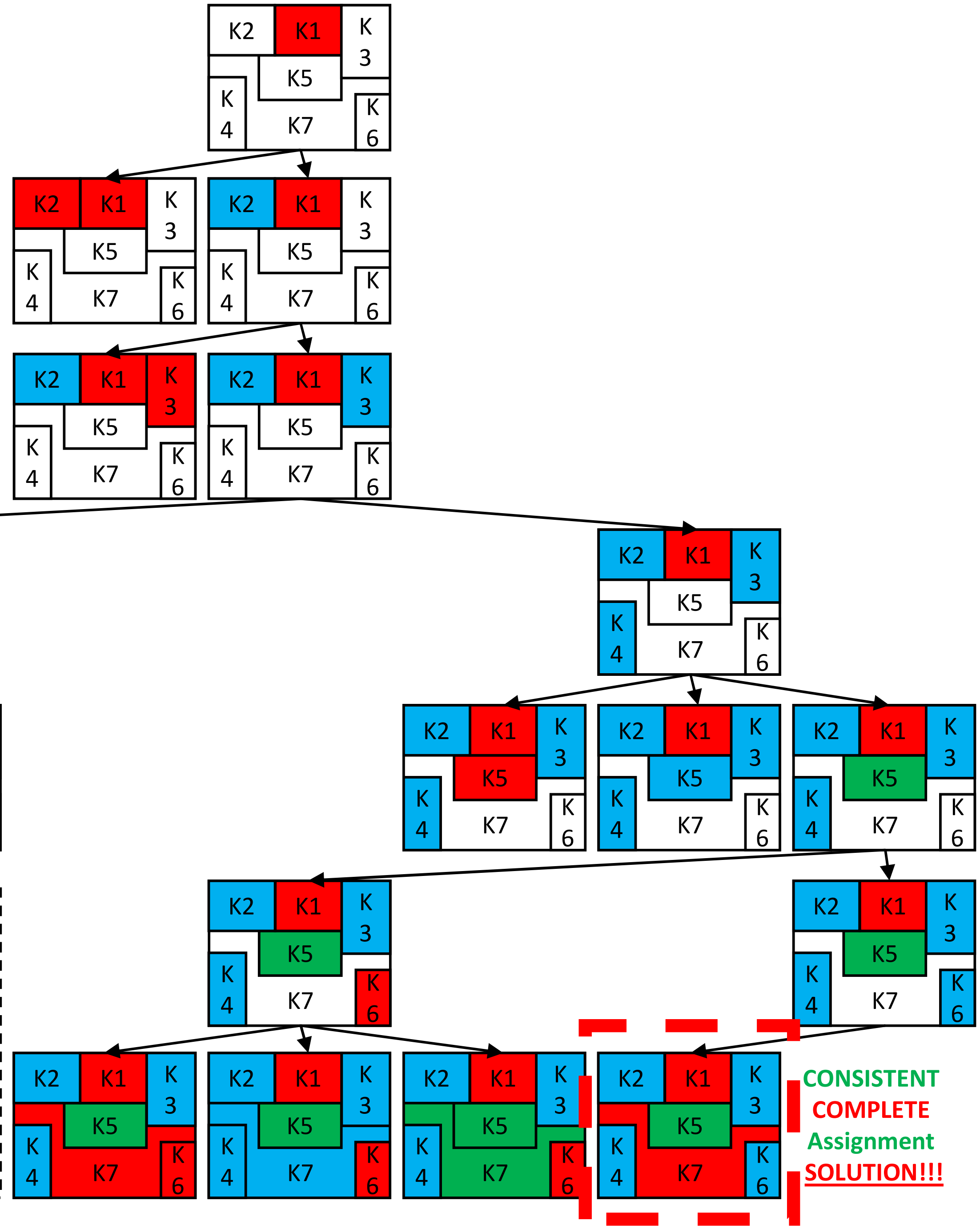
CSP Backtracking: Pseudocode

function BACKTRACKING-SEARCH(*csp*) **returns** a solution or *failure*
  return BACKTRACK(*csp*, { })

function BACKTRACK(*csp*, *assignment*) **returns** a solution or *failure*
  if *assignment* is complete **then return** *assignment*
  var ← SELECT-UNASSIGNED-VARIABLE(*csp*, *assignment*)
  for each *value* **in** ORDER-DOMAIN-VALUES(*csp*, *var*, *assignment*) **do**
    if *value* is consistent with *assignment* **then**
      add {*var* = *value*} to *assignment*
      inferences ← INFERENCE(*csp*, *var*, *assignment*)
      if *inferences* ≠ *failure* **then**
        add *inferences* to *csp*
        result ← BACKTRACK(*csp*, *assignment*)
        if *result* ≠ *failure* **then return** *result*
        remove *inferences* from *csp*
      remove {*var* = *value*} from *assignment*
  return *failure*

Which variable should we choose to assign a value to next?
Does it matter?

Variable assignment order:
K1, K2, K3, K4, K5, K6, K7
(but do we have to keep that order?)



K1 = ???

K2 = ???

K3 = ???

K4 = ???

K5 = ???

K6 = ???

K7 = ???

Visited / dead ends:
Complete, but inconsistent assignments

Variable assignment order: K1, K2, K3, K4, K5, K6, K7

Variable Ordering: Alternatives

function BACKTRACKING-SEARCH(*csp*) **returns** a solution or *failure*
  return BACKTRACK(*csp*, { })

function BACKTRACK(*csp*, *assignment*) **returns** a solution or *failure*
  if *assignment* is complete **then return** *assignment*
  var ← SELECT-UNASSIGNED-VARIABLE(*csp*, *assignment*)
  for each *value* **in** ORDER-DOMAIN-VALUES(*csp*, *var*, *assignment*) **do**
    if *value* is consistent with *assignment* **then**
      add {*var* = *value*} to *assignment*
      inferences ← INFERENCE(*csp*, *var*, *assignment*)
      if *inferences* ≠ *failure* **then**
        add *inferences* to *csp*
        result ← BACKTRACK(*csp*, *assignment*)
        if *result* ≠ *failure* **then return** *result*
        remove *inferences* from *csp*
      remove {*var* = *value*} from *assignment*
  return *failure*

You can modify this function to change the variable ordering and potentially improve performance

Variable Ordering: Alternatives

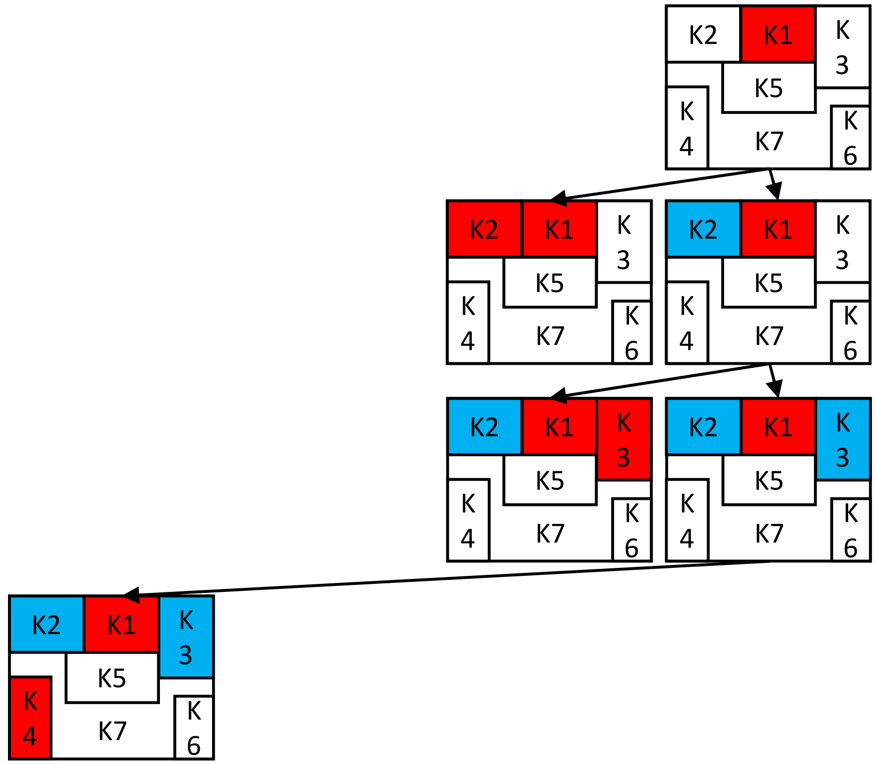
CSP Backtracking algorithm can use a number of variable ordering strategies:

- **Static: choose the variables in order (we did that)**
- **Random: order variables in random sequence**
- **Minimum-remaining-values (MRV) heuristic:**
  - **choose the variable with the "fewest" legal values**
- **Degree heuristic:**
  - **choose the variable involved in the largest amount of constraints on other unassigned variables**
  - **choose the variable with highest node degree on a constraint graph**

Variable Ordering: MRV Heuristic

As CSP Backtracking algorithm progresses, the number of possible value assignments for each variable will shrink (due to constraints):

- MRV uses “fail-first” heuristics (also called “most constrained variable” heuristics)
- MRV picks a variable with lowest value assignment options “left”
  - expecting to limit exploration depth
  - likely to find a failure assignment faster
- Usually better than static and random orderings on average



K1 = ???

K2 = ???

K3 = ???

K4 = ???

Which variable to explore next (ignore the EXPECTED sequence on the right)?
Available options:

K5: {GREEN}

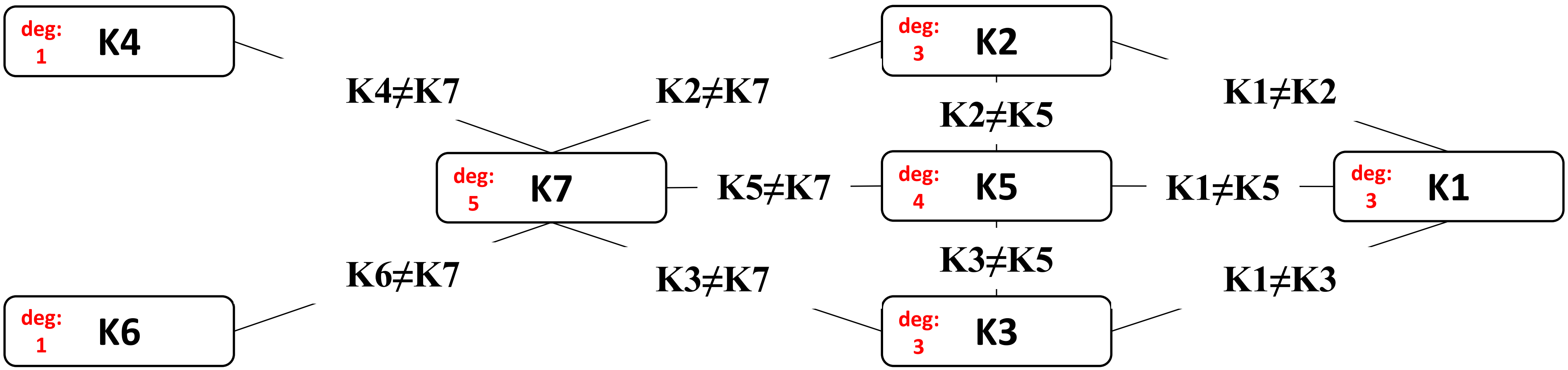
K6: {RED, BLUE, GREEN}

K7: {GREEN}

MRV should pick K5 or K7 (“fail first” variable).
Tie needs to be resolved.

Variable Ordering: Degree Heuristics

Consider the following constraint graph representation of the problem we analyzed:



- degree heuristics is considered less effective than MRV
- degree heuristics can be used as a tie-breaker (two variables with the same "potential" according to MRV)
- attempts to reduce the branching factor on future choices

Value Ordering: Alternatives

```
function BACKTRACKING-SEARCH(csp) returns a solution or failure
   return BACKTRACK(csp, { })

function BACKTRACK(csp, assignment) returns a solution or failure
   if assignment is complete then return assignment
   var ← SELECT-UNASSIGNED-VARIABLE(csp, assignment)
   for each value in ORDER-DOMAIN-VALUES(csp, var, assignment) do
      if value is consistent with assignment then
         add {var = value} to assignment
         inferences ← INFERENCE(csp, var, assignment)
         if inferences ≠ failure then
            add inferences to csp
            result ← BACKTRACK(csp, assignment)
            if result ≠ failure then return result
            remove inferences from csp
         remove {var = value} from assignment
   return failure
```

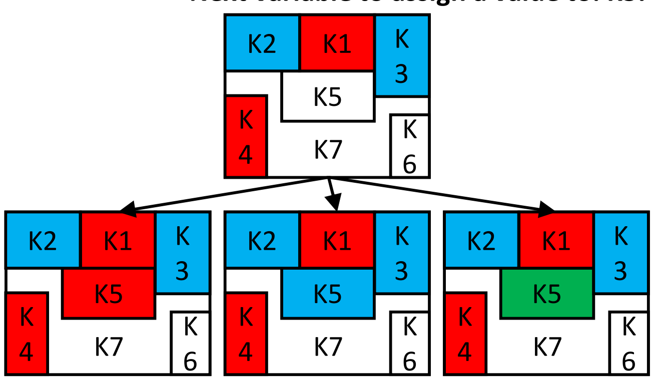
You can modify this order to change the value assignment ordering and potentially improve performance

Least-Constraining-Value Heuristics

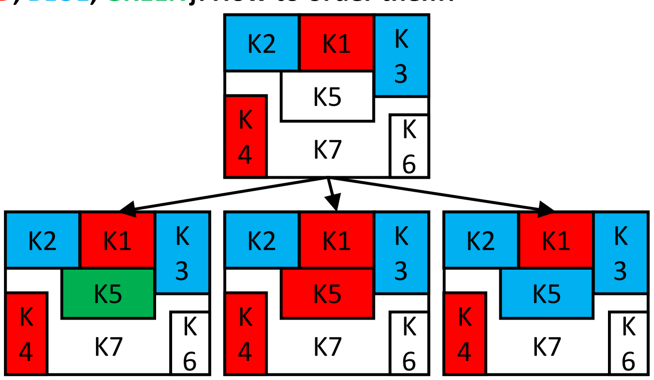
We picked (SELECT-UNASSIGNED-VARIABLE) the next variable to assign a value to and we have a number of values to choose from. What next?

- use the least-constraining-value heuristic
  - picks a value that rules out the fewest choices for neighboring variables in the constraing graph (increase flexibility for FUTURE assignments)
  - ORDER-DOMAIN-VALUES is the function that orders values here

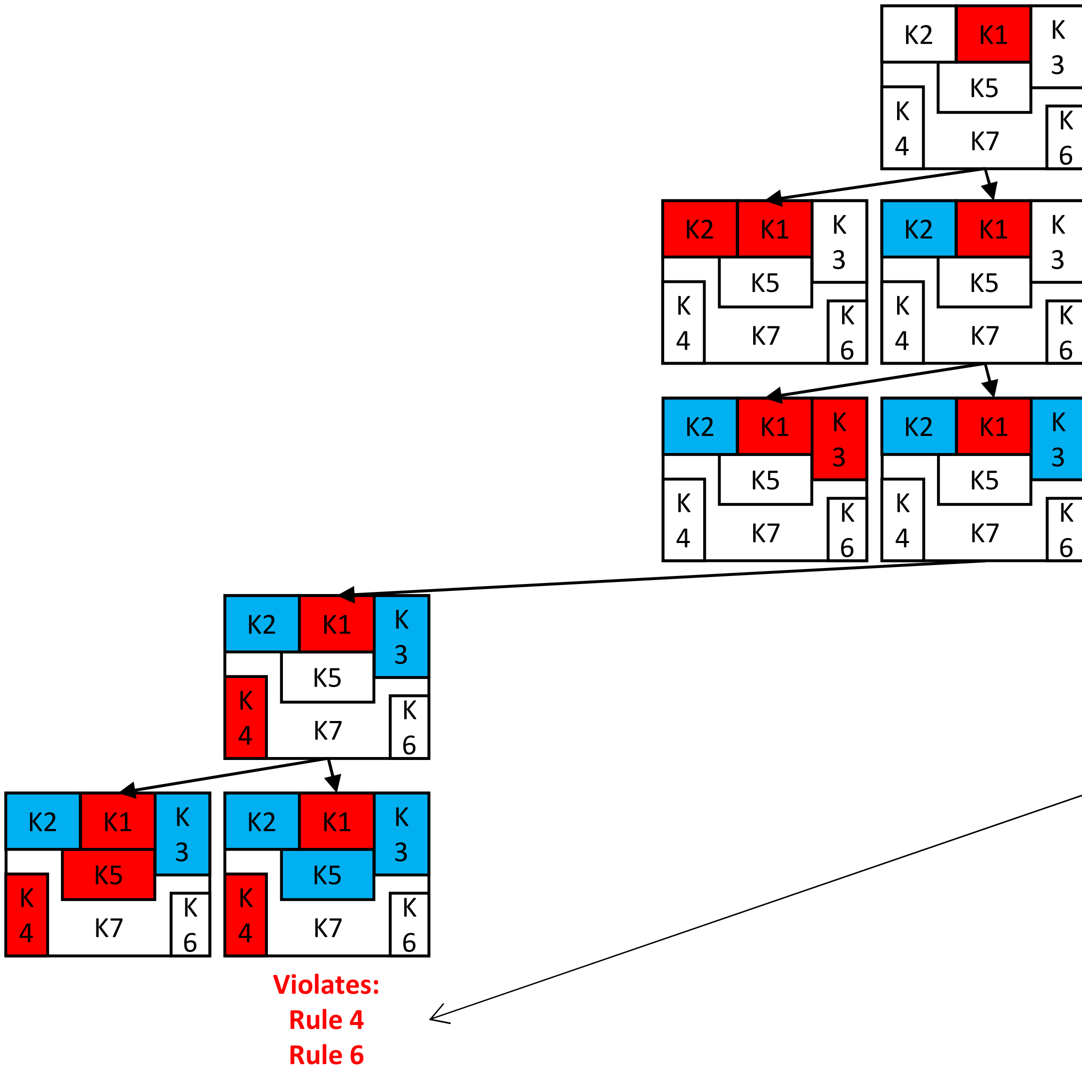
Next variable to assign a value to: K5. Three options available: {RED, BLUE, GREEN}. How to order them?



STATIC value ordering:
poor, best option is last



LEAST-CONSTRAINING-VALUE value ordering:
better, best option is first



INCONSISTENCY of this PARTIAL assignment was established at the node level:

node consistency

Can we detect / "predict" INCONSISTENCIES earlier and prune useless branches? YES!

Variable assignment order: K1, K2, K3, K4, K5, K6, K7 | **Value assignment order: RED, BLUE, GREEN**