Blood Transfusion Service via PDDL and IndiGolog

Planning & Reasoning Project

Francesca Andreotti - 1696976

Sapienza University of Rome

Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

Sapienza
Università di Roma

Contents

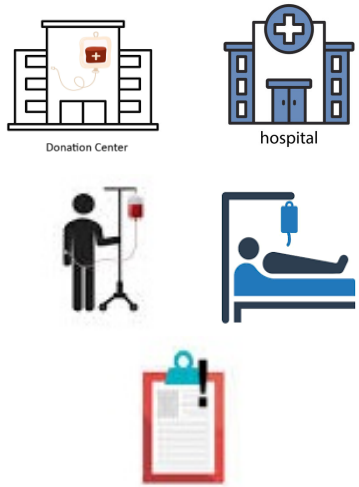
Blood Transfusion Service - Overview

Idea

Blood bags management system ensuring eligible **donations** and compatible **blood delivery** and **transfusion**, while minimizing transport costs and prioritizing urgent transfusions

Key properties

- **Donors/patients features** and **locations**
  $\rightarrow$ centers, hospitals
- **Donor eligibility**, based on *physical health criteria*
  $\rightarrow$ age, blood pressure, hemoglobin level
- **Blood type classification**
  ABO blood group: A, B, AB, or O
  Rh factor: positive, negative
- **Compatibility blood criteria** for safe transfusions
- **Blood bag availability** monitored per blood type
- **Prioritized urgent transfusions** over stable cases



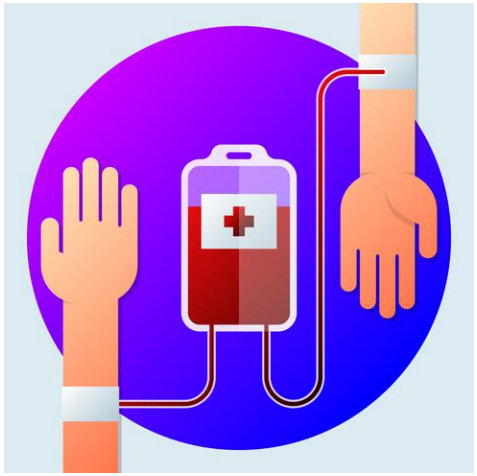
Goal and Optimization

Goal

All patients receive compatible blood transfusions

AND

All donors are screened, allowing all eligible donors to proceed for donating



Optimization objectives

- Minimize cumulative **action costs**
- Minimize travel **distances** between centers and hospitals
- Prioritize **urgent transfusions**

PDDL Domain - Predicates and Functions

Types

```
(:types
   blood-carrier
   donor patient - blood-carrier
   location
   hospital center - location
   bloodgroup
   rhf
)
```

Functions

```
(:functions
   ;; donor physical parameters
   (donor-age ?d - donor)
   (max-pressure ?d - donor)
   (min-pressure ?d - donor)
   (hemoglobin ?d - donor)

   ;; available blood units at a location
   (available-bags ?bg - bloodgroup ?rh - rhf ?l - location)
   (supplies ?bg - bloodgroup ?rh - rhf ?h - hospital)

   ;; distance and total cost
   (distance ?from - center ?to - hospital)
   (total-cost)
   (urgency-penalty)
)
```

Predicates

```
(:predicates
   ;; locations
   (donor-at ?d - donor ?c - center)
   (patient-at ?p - patient ?h - hospital)

   ;; blood type properties
   (has-bloodgroup ?x - blood-carrier ?bg - bloodgroup)
   (has-rhf ?x - blood-carrier ?rh - rhf)

   ;; donor status
   (accepted ?d - donor)
   (rejected ?d - donor)
   (donated ?d - donor)

   ;; patient status
   (needs-transfusion ?p - patient)
   (urgent ?p - patient)
   (bag-assigned ?p - patient)

   ;; blood compatibility predicates
   (rh-compatible ?prhf ?brhf - rhf)
   (abo-compatible ?pabo ?babo - bloodgroup)
)
```

PDDL Domain - Actions

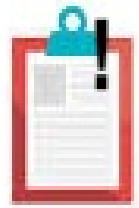
```
(:action accept
  :parameters (?d - donor)
  :precondition (and
    (>= (donor-age ?d) 18)
    (<= (donor-age ?d) 65)
    (>= (hemoglobin ?d) 12.5)
    (>= (max-pressure ?d) 90)
    (<= (max-pressure ?d) 140)
    (>= (min-pressure ?d) 60)
    (<= (min-pressure ?d) 90)
  )
  :effect (and
    (accepted ?d)
    (increase (total-cost) 400)
  )
)
```

```
(:action reject
  :parameters (?d - donor)
  :precondition (and (or
    (< (donor-age ?d) 18)
    (> (donor-age ?d) 65)
    (< (hemoglobin ?d) 12.5)
    (< (max-pressure ?d) 90)
    (> (max-pressure ?d) 140)
    (< (min-pressure ?d) 60)
    (> (min-pressure ?d) 90))
  )
  :effect (and
    (rejected ?d)
    (increase (total-cost) 500)
  )
)
```

Eligibility checks

Accept Donor

Verify donor eligibility for donating blood, accepting eventually the donation request.

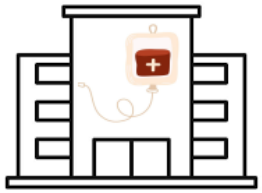


Reject Donor

Verify donor ineligibility according to the physical parameters, rejecting eventually the donation request.

PDDL Domain - Actions

```
(:action donate
  :parameters (?d - donor ?bg - bloodgroup ?rh - rhf ?c - center)
  :precondition (and
    (donor-at ?d ?c)
    (not (donated ?d))
    (accepted ?d)
    (has-bloodgroup ?d ?bg)
    (has-rhf ?d ?rh)
  )
  :effect (and
    (donated ?d)
    (increase (available-bags ?bg ?rh ?c) 1)
    (increase (total-cost) 200)
  )
)
(:action moveto
  :parameters (?from - center ?to - hospital
               ?bg - bloodgroup ?rh - rhf ?p - patient)
  :precondition (and
    (> (available-bags ?bg ?rh ?from) 0)
    (patient-at ?p ?to)
    (not (bag-assigned ?p))
    (needs-transfusion ?p)
    (exists (?pbg - bloodgroup ?prh - rhf)
      (and
        (has-bloodgroup ?p ?pbg)
        (has-rhf ?p ?prh)
        (abo-compatible ?pbg ?bg)
        (rh-compatible ?prh ?rh)
      )
  ))
  :effect (and
    (bag-assigned ?p)
    (decrease (available-bags ?bg ?rh ?from) 1)
    (increase (available-bags ?bg ?rh ?to) 1)
    (increase (total-cost) (* 10 (distance ?from ?to)))
  )
```



Donation Center

Blood bags logistics

Donate Blood

Execute blood donation by an accepted donor who has not donated

Move blood bags

Move a compatible blood bag assigned to a patient from a center to the hospital where the patient is located.

PDDL Domain - Actions

Handling transfusions and emergencies

Perform transfusion

Execute a standard blood transfusion

```
(:action transfuse
  :parameters (?p - patient ?h - hospital ?bg - bloodgroup
               ?rh - rhf ?pbg - bloodgroup ?prh - rhf)
  :precondition (and
    (not (urgent ?p))
    (patient-at ?p ?h)
    (needs-transfusion ?p)
    (has-bloodgroup ?p ?pbg)
    (has-rhf ?p ?prh)
    (abo-compatible ?pbg ?bg)
    (rh-compatible ?prh ?rh)
    (> (available-bags ?bg ?rh ?h) 0)
  )
  :effect (and
    (not (needs-transfusion ?p))
    (decrease (available-bags ?bg ?rh ?h) 1)
    (increase (urgency-penalty) 20)
    (increase (total-cost) 100)
  )
```

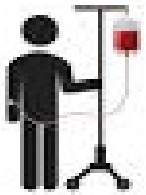
Perform urgent transfusion

Transfuse urgently critical cases

```
(:action transfuse-urgently
  :parameters (?p - patient ?h - hospital ?bg - bloodgroup
               ?rh - rhf ?pbg - bloodgroup ?prh - rhf)
  :precondition (and
    (urgent ?p)
    (patient-at ?p ?h)
    (needs-transfusion ?p)
    (has-bloodgroup ?p ?pbg)
    (has-rhf ?p ?prh)
    (abo-compatible ?pbg ?bg)
    (rh-compatible ?prh ?rh)
    (> (available-bags ?bg ?rh ?h) 0)
  )
  :effect (and
    (not (needs-transfusion ?p))
    (decrease (available-bags ?bg ?rh ?h) 1)
    (increase (urgency-penalty) 0)
  )
```

PDDL Problems - Initial State

Donor Information



```
(donor-at donor1 center1)
(donor-at donor2 center1)
(donor-at donor3 center1)
(donor-at donor4 center2)
...

(has-bloodgroup donor1 A)
(has-bloodgroup donor2 O)
(has-bloodgroup donor3 AB)
(has-bloodgroup donor4 B)
...

(has-rhf donor1 pos)
(has-rhf donor2 neg)
(has-rhf donor3 pos)
(has-rhf donor4 pos)
...
```

Donors Conditions

```
(= (donor-age donor1) 30)
(= (donor-age donor2) 45)
(= (donor-age donor3) 40)
(= (donor-age donor4) 48)
...

(= (hemoglobin donor1) 13.0)
(= (hemoglobin donor2) 14.0)
(= (hemoglobin donor3) 15.0)
(= (hemoglobin donor4) 13.0)
...

(= (max-pressure donor1) 190)
(= (min-pressure donor1) 80)
(= (max-pressure donor2) 110)
(= (min-pressure donor2) 70)
(= (max-pressure donor3) 120)
...
```

Patients Information

```
(patient-at patient1 hospital1)
(patient-at patient2 hospital1)
(patient-at patient3 hospital2)
(patient-at patient4 hospital2)
...

(has-bloodgroup patient1 O)
(has-bloodgroup patient2 AB)
(has-bloodgroup patient3 B)
(has-bloodgroup patient4 O)
...

(has-rhf patient1 pos)
(has-rhf patient2 pos)
(has-rhf patient3 neg)
(has-rhf patient4 neg)
...

(needs-transfusion patient1)
(needs-transfusion patient2)
(needs-transfusion patient3)
(needs-transfusion patient4)
```

PDDL Problems - Initial State

Blood bag supplies

```
(= (available-bags A pos center1) 0)
(= (available-bags A neg center1) 0)
(= (available-bags B pos center1) 0)
(= (available-bags B neg center1) 0)
(= (available-bags AB pos center1) 0)
(= (available-bags AB neg center1) 0)
(= (available-bags O pos center1) 0)
(= (available-bags O neg center1) 0)
...

(= (available-bags A pos hospital1) 0)
(= (available-bags A neg hospital1) 1)
(= (available-bags B pos hospital1) 0)
(= (available-bags B neg hospital1) 1)
(= (available-bags AB pos hospital1) 3)
(= (available-bags AB neg hospital1) 0)
(= (available-bags O pos hospital1) 0)
(= (available-bags O neg hospital1) 0)
...

(= (distance center1 hospital1) 10)
(= (distance center1 hospital2) 30)
(= (distance center1 hospital3) 45)
...
```

Blood Compatibility

```
(abo-compatible A A)
(abo-compatible A O)
(abo-compatible O O)
(abo-compatible B B)
(abo-compatible B O)

(abo-compatible AB A)
(abo-compatible AB B)
(abo-compatible AB AB)
(abo-compatible AB O)

(rh-compatible pos pos)
(rh-compatible neg neg)
(rh-compatible pos neg)
```

PDDL Problems - Goal and Metrics

Costs Initialization

```
(= (total-cost) 0)
(= (urgency-penalty) 0)
```

Goal State & Metrics

```
(:goal
  (and
    (forall (?p - patient)
      (not (needs-transfusion ?p))
    )
    (forall (?d - donor)
      (or
        (and (accepted ?d) (donated ?d) (not (rejected ?d)))
        (and (not (accepted ?d)) (rejected ?d))
    ))
  )
)

(:metric minimize (+ (total-cost)(urgency-penalty)))
```

Planner & Search Heuristics - ENHSP

ENHSP Automated planning system

- Input: PDDL domain and PDDL problem files
- Translation of the input into a **graph-search** problem
  - → Nodes as planner states
  - → Forward expansion of the graph
  - → Heuristic-guided search toward the goal
- Output: Solution plan (if one exists) along with associate metrics

Configurations

sat-hadd

Search: Greedy Best-First Search

Heuristic: hadd

Goal: Find quickly valid plan

→not necessarily optimal

opt-blind

Search: A* Search

Heuristic: blind

Goal: Find optimal plan

→very slow

opt-hmax

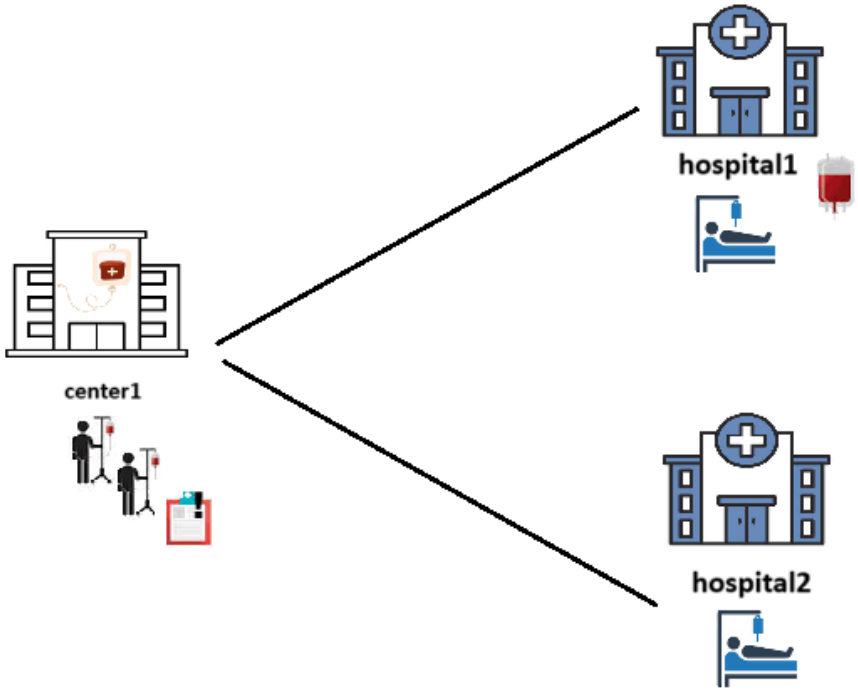
Search: A* Search

Heuristic: hmax

Goal: Find cost-optimal plan

→guided search

Problem 1 - Basic Scenario



Initial state

One center, two hospitals
Two donors, one not eligible
Two patients
A compatible blood bag
Same urgency
Same distances

```
(:objects
   donor1 donor2 - donor
   patient1 patient2 - patient
   center1 - center
   hospital1 hospital2 - hospital
   A B AB O - bloodgroup
   pos neg - rhf
 )
```

Metric

→ Minimize the sum of action costs

Basic Scenario - Output Plans & Metrics

Metric	sat-hadd	opt-blind	opt-hmax
Plan-Length	6	6	6
Plan Cost	2300	2300	2300
Planning Time	196	183	190
Heuristic Time	2	0	2
Search Time	8	5	8
Expanded Nodes	7	20	15

sat-hadd

```
(accept donor1 )
(reject donor2 )
(donate donor1 A neg center1 )
(moveto center1 hospital2 A neg patient2 )
(transfuse patient1 hospital1 0 pos B pos )
(transfuse patient2 hospital2 A neg AB neg )
```

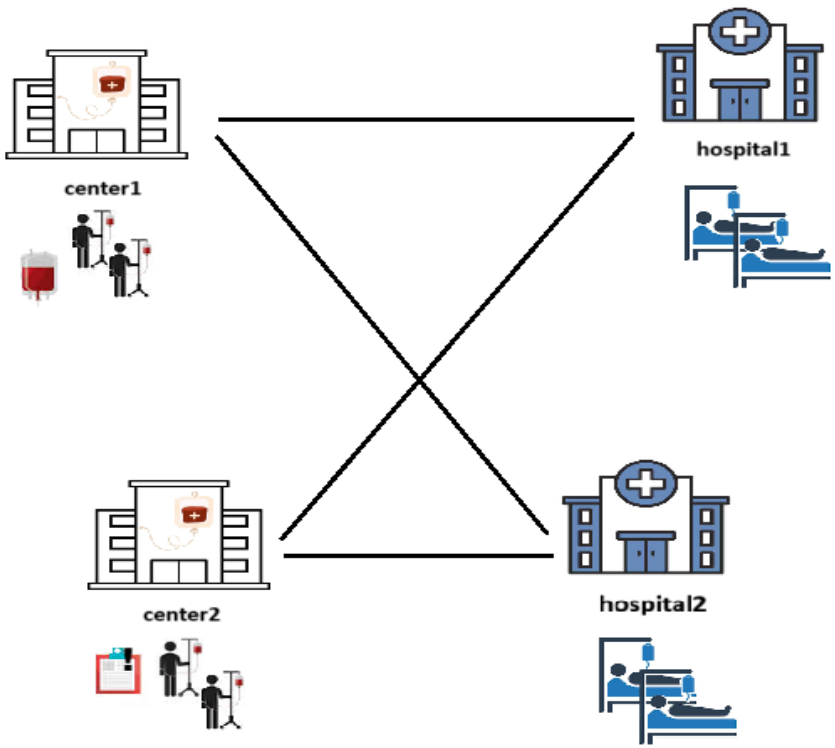
opt-blind

```
(transfuse patient1 hospital1 0 pos B pos )
(accept donor1 )
(donate donor1 A neg center1 )
(moveto center1 hospital2 A neg patient2 )
(transfuse patient2 hospital2 A neg AB neg )
(reject donor2 )
```

opt-hmax

```
(accept donor1 )
(donate donor1 A neg center1 )
(moveto center1 hospital2 A neg patient2 )
(reject donor2 )
(transfuse patient1 hospital1 0 pos B pos )
(transfuse patient2 hospital2 A neg AB neg )
```

Problem 2 - Transport Costs Minimization



Initial state

Two centers, two hospitals
Four donors
Four patients
A blood bag available
Same urgency
Different distances values

Metric

$\rightarrow$ Minimize sum of actions cost and **transport cost** of blood bags

Problem 2 - Output Plan & Metrics

Metric	sat-hadd	opt-blind	opt-hmax
Plan-Length	15	15	15
Plan Cost	4000	4000	4000
Planning Time	257	313	390
Heuristic Time	11	0	76
Search Time	27	103	142
Expanded Nodes	16	3538	1807

sat-hadd

```
(accept donor1 )
(accept donor2 )
(accept donor4 )
(reject donor3 )
(donate donor1 A pos center1 )
(donate donor2 0 neg center1 )
(donate donor4 0 pos center2 )
(moveto center1 hospital1 0 neg patient1 )
(transfuse patient1 hospital1 0 neg B neg )
(moveto center2 hospital2 0 pos patient3 )
 (transfuse patient3 hospital2 0 pos B pos )
 (moveto center1 hospital2 A pos patient2 )
 (transfuse patient2 hospital2 A pos A pos )
 (moveto center1 hospital1 0 neg patient4 )
 (transfuse patient4 hospital1 0 neg 0 neg )
```

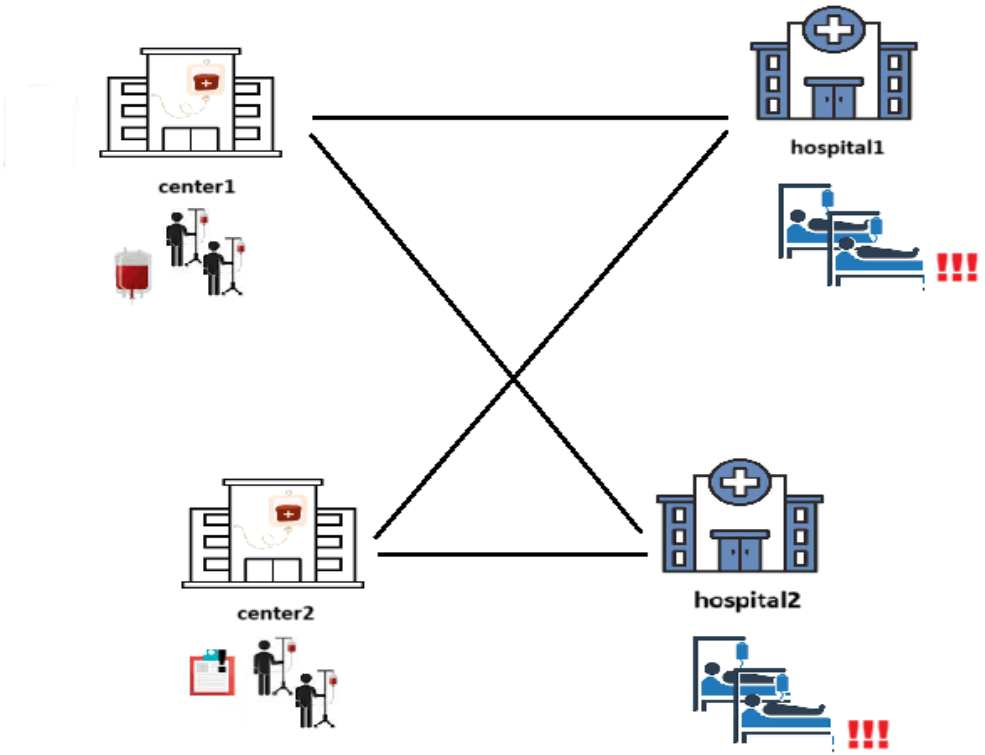
opt-blind

```
(moveto center1 hospital1 0 neg patient1 )
(transfuse patient1 hospital1 0 neg B neg )
(accept donor4 )
(donate donor4 0 pos center2 )
(moveto center2 hospital2 0 pos patient3 )
(transfuse patient3 hospital2 0 pos B pos )
(accept donor1 )
(donate donor1 A pos center1 )
(moveto center1 hospital2 A pos patient2 )
(accept donor2 )
 (donate donor2 0 neg center1 )
 (moveto center1 hospital1 0 neg patient4 )
 (transfuse patient4 hospital1 0 neg 0 neg )
 (transfuse patient2 hospital2 A pos A pos )
 (reject donor3 )
```

opt-hmax

```
(accept donor1 )
(donate donor1 A pos center1 )
(moveto center1 hospital2 A pos patient2 )
(transfuse patient2 hospital2 A pos A pos )
(accept donor4 )
(donate donor4 0 pos center2 )
(moveto center2 hospital2 0 pos patient3 )
(transfuse patient3 hospital2 0 pos B pos )
(accept donor2 )
(reject donor3 )
 (donate donor2 0 neg center1 )
 (moveto center1 hospital1 0 neg patient4 )
 (transfuse patient4 hospital1 0 neg 0 neg )
 (moveto center1 hospital1 0 neg patient1 )
 (transfuse patient1 hospital1 0 neg B neg )
```

Problem 3 - Urgencies Prioritization




```
(urgent patient1)
(urgent patient4)
```

Initial state

Two centers, two hospitals
Four donors
Four patients
A blood bag available
Urgent patients
no supplies available
Different distances values

Metric

→ Minimize sum of actions cost and transport cost of blood bags

→ **Prioritize urgent transfusions** through penalty

Problem 3 - Output Plan & Metrics

Metric	sat-hadd	opt-blind	opt-hmax
Plan-Length	15	15	15
Plan Cost	3240	3240	3240
Planning Time	281	346	529
Heuristic Time	14	0	111
Search Time	30	117	220
Expanded Nodes	16	3789	2209

sat-hadd

```
(accept donor1 )
(accept donor2 )
(accept donor3 )
(reject donor4 )
(donate donor1 A pos center1 )
(donate donor2 0 neg center1 )
(donate donor3 0 neg center2 )
(moveto center1 hospital1 0 neg patient1 )
(transfuse patient1 hospital1 0 neg B neg )
(moveto center2 hospital1 0 neg patient4 )
 (moveto center1 hospital2 0 neg patient3 )
 (transfuse patient3 hospital2 0 neg B pos )
 (moveto center1 hospital2 A pos patient2 )
 (transfuse-urgently patient4 hospital1 0 neg 0 neg
 (transfuse-urgently patient2 hospital2 A pos A pos
```

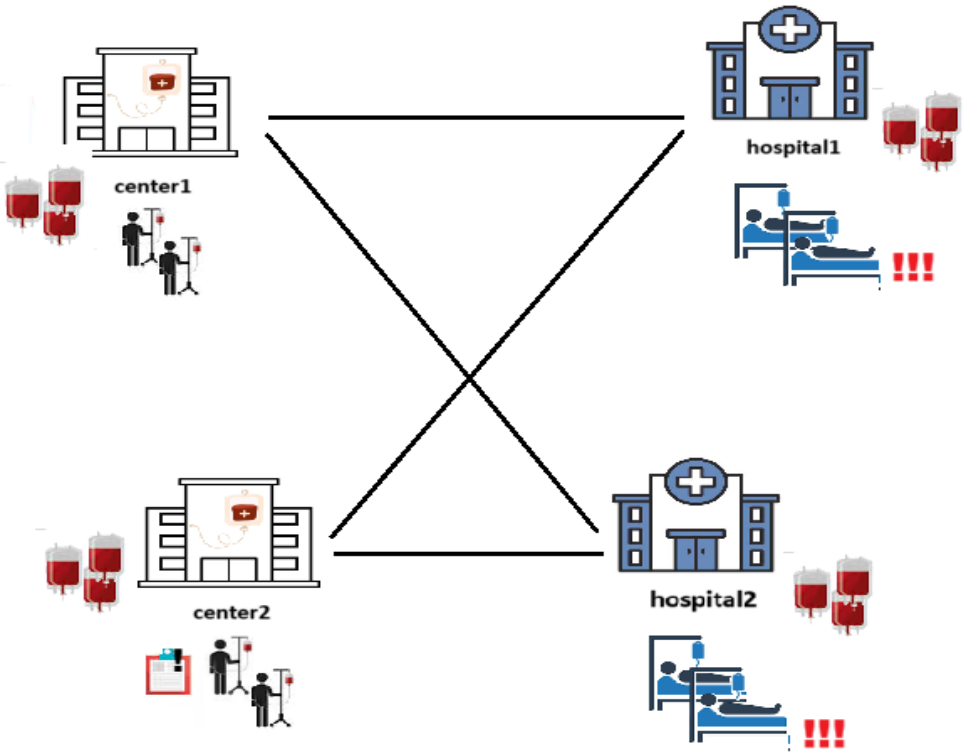
opt-blind

```
(accept donor1 )
(donate donor1 A pos center1 )
(moveto center1 hospital2 A pos patient2 )
(transfuse-urgently patient2 hospital2 A pos A pos
(moveto center1 hospital1 0 neg patient4 )
(transfuse-urgently patient4 hospital1 0 neg 0 neg
(accept donor2 )
(donate donor2 0 neg center1 )
(moveto center1 hospital2 0 neg patient3 )
(transfuse patient3 hospital2 0 neg B pos )
: (accept donor3 )
: (donate donor3 0 neg center2 )
: (moveto center2 hospital1 0 neg patient1 )
(transfuse patient1 hospital1 0 neg B neg )
(reject donor4 )
```

opt-hmax

```
(accept donor1 )
(donate donor1 A pos center1 )
(moveto center1 hospital2 A pos patient2 )
(transfuse-urgently patient2 hospital2 A pos A pos
(moveto center1 hospital1 0 neg patient4 )
(transfuse-urgently patient4 hospital1 0 neg 0 neg
(accept donor2 )
(donate donor2 0 neg center1 )
(moveto center1 hospital2 0 neg patient3 )
(accept donor3 )
 (donate donor3 0 neg center2 )
 (reject donor4 )
 (moveto center2 hospital1 0 neg patient1 )
 (transfuse patient1 hospital1 0 neg B neg )
 (transfuse patient3 hospital2 0 neg B pos )
```

Problem 4 - Blood Bags Supplies



Initial state

Two centers, two hospitals
Four donors
Four patients
A blood bag available
Urgent patients

Supplies available

Different distances values

```
(= (supplies 0 pos hospital1) 1)
(= (supplies 0 neg hospital1) 1)

(= (supplies 0 pos hospital2) 0)
(= (supplies 0 neg hospital2) 1)
```

Metric

→ Minimize sum of actions cost and transport cost of blood bags

→ Prioritize urgent transfusions through penalty

PDDL Domain - Emergency Supplies Distribution

```
(:action transfuse-urgently-supply
  :parameters (?p - patient ?h - hospital ?bg - bloodgroup
               ?rh - rhf ?pbg - bloodgroup ?prh - rhf)
  :precondition (and
    (urgent ?p)
    (patient-at ?p ?h)
    (needs-transfusion ?p)
    (has-bloodgroup ?p ?pbg)
    (has-rhf ?p ?prh)
    (abo-compatible ?pbg ?bg)
    (rh-compatible ?prh ?rh)
    (> (supplies ?bg ?rh ?h) 0)
    (<= (available-bags ?bg ?rh ?h) 0)
  )
```

```
:effect (and
  (not (needs-transfusion ?p))
  (decrease (supplies ?bg ?rh ?h) 1)
  (increase (urgency-penalty) 0)
  (increase (total-cost) 100)
 )
```

Emergency supply usage

Execute urgent transfusions using supplies already stored in the hospital

→ Supplies **limited** to 0 blood group

→ **More expensive** transfusion wrt standard urgent ones.

Problem 4 - Output Plan & Metrics

Metric	sat-hadd	opt-blind	opt-hmax
Plan-Length	13	14	13
Plan Cost	3140	3040	3040
Planning Time	274	369	445
Heuristic Time	13	0	105
Search Time	29	119	205
Expanded Nodes	14	4335	3183

sat-hadd

```
(accept donor3 )
(accept donor1 )
(accept donor2 )
(donate donor3 O neg center2 )
(reject donor4 )
(donate donor1 A pos center1 )
(donate donor2 B neg center1 )
(transfuse-urgently-supply patient2 hospital2 A pos A pos
(moveto center1 hospital2 B pos patient3 )
(transfuse patient3 hospital2 B pos B pos )
 (moveto center2 hospital1 O neg patient1 )
 (transfuse patient1 hospital1 O neg B neg )
 (transfuse-urgently-supply patient4 hospital1 O neg O neg
```

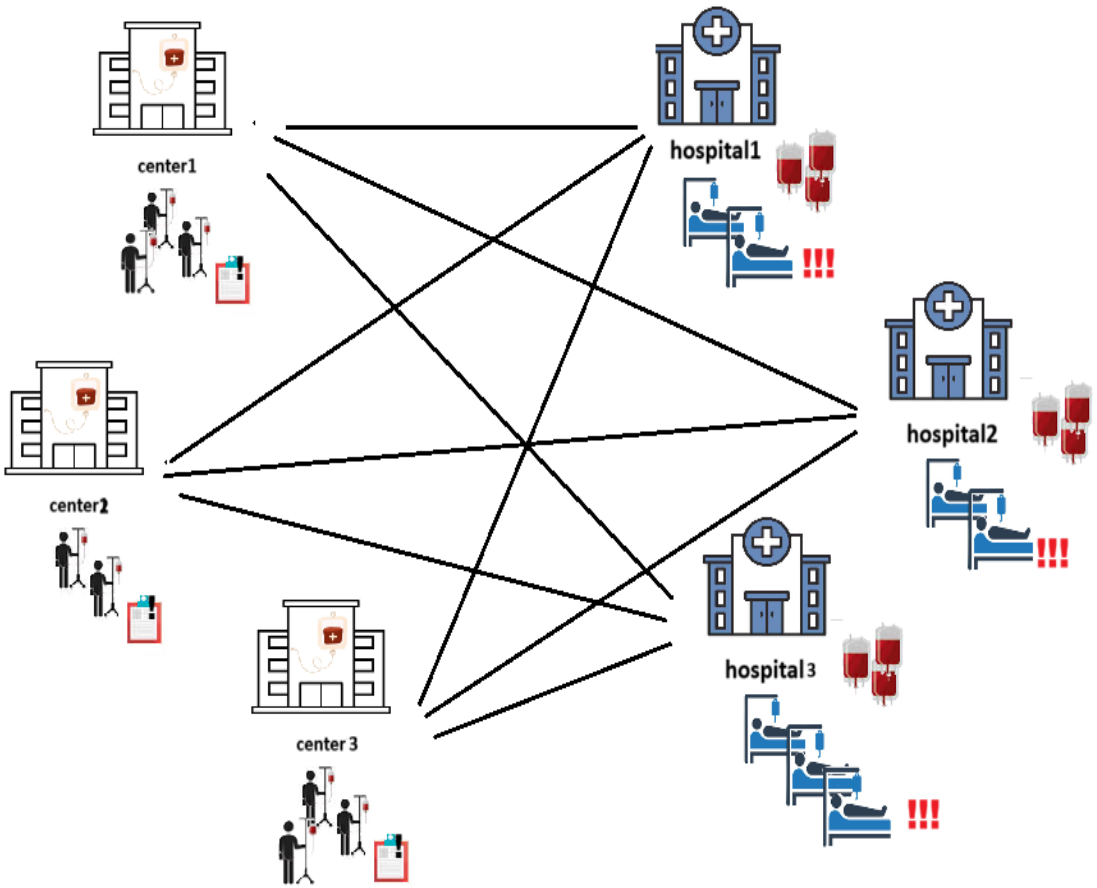
opt-blind

```
(accept donor1 )
(donate donor1 A pos center1 )
(moveto center1 hospital2 A pos patient2 )
(transfuse-urgently patient2 hospital2 A pos A pos )
(moveto center1 hospital2 B pos patient3 )
(transfuse-urgently-supply patient4 hospital1 O neg O neg
(accept donor2 )
(donate donor2 B neg center1 )
(moveto center1 hospital1 B neg patient1 )
(transfuse patient1 hospital1 B neg B neg )
 (transfuse patient3 hospital2 B pos B pos )
 (accept donor3 )
 (donate donor3 O neg center2 )
 (reject donor4 )
```

opt-hmax

```
(moveto center1 hospital2 B pos patient3 )
(accept donor3 )
(donate donor3 O neg center2 )
(accept donor1 )
(accept donor2 )
(reject donor4 )
(donate donor1 A pos center1 )
(donate donor2 B neg center1 )
(moveto center1 hospital1 B neg patient1 )
(transfuse patient1 hospital1 B neg B neg )
 (transfuse patient3 hospital2 B pos B pos )
 (transfuse-urgently-supply patient2 hospital2 A pos A pos
 (transfuse-urgently-supply patient4 hospital1 O neg O neg
```

Problem 5 - Advanced Scenario



Initial state

Three centers, **three** hospitals
Eight donors
Seven patients
More blood bags available
Urgent patients

Metric

→ Minimize sum of actions cost and transport cost of blood bags

→ Prioritize urgent transfusions through penalty

Problem 5 - Output Plan & Metrics

Metric	sat-hadd	opt-blind	opt-hmax
Planning Time	436	150035	68967
Heuristic Time	44	440	33622
Search Time	93	150035	68737
Expanded Nodes	24	7937574	3286070

sat-hadd

```
(accept donor2 )
(accept donor3 )
(accept donor4 )
(accept donor6 )
(accept donor7 )
(donate donor2 O neg center1 )
(reject donor1 )
(reject donor5 )
(reject donor8 )
(donate donor3 A pos center1 )
 (donate donor4 B pos center2 )
 (donate donor6 O pos center3 )
 (donate donor7 AB neg center3 )
 (transfuse patient6 hospital3 AB neg AB neg )
 (transfuse patient3 hospital2 B neg B neg )
 (transfuse patient2 hospital1 B neg AB neg )
 (transfuse patient5 hospital3 A neg A neg )
 (moveto center3 hospital3 A neg patient7 )
 (transfuse-urgently patient7 hospital3 A neg A pos )
 (moveto center3 hospital1 O pos patient1 )
 (moveto center1 hospital2 O neg patient4 )
 (transfuse-urgently patient1 hospital1 O pos O pos )
 (transfuse-urgently patient4 hospital2 O neg O neg )
```

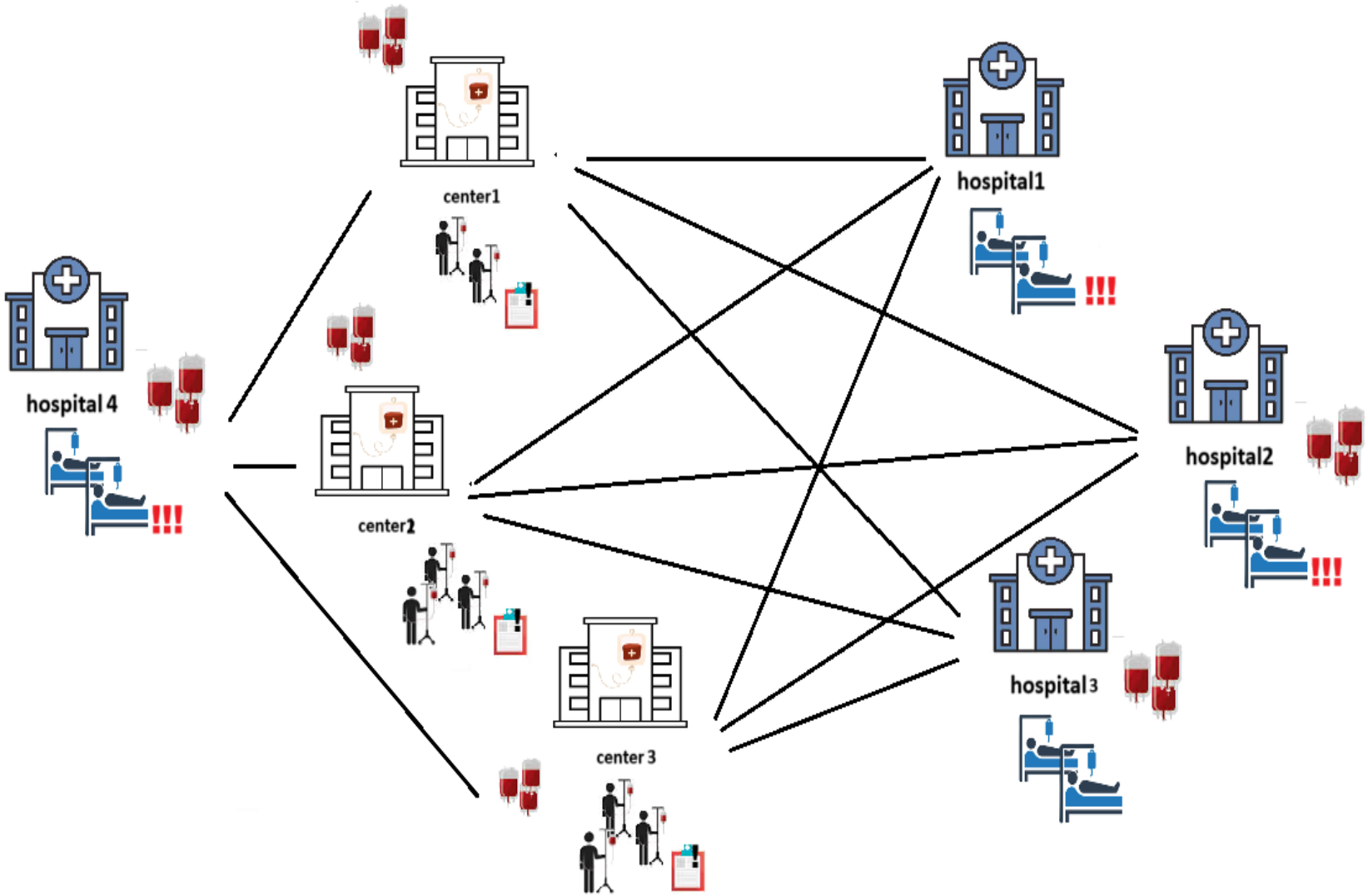
opt-blind

```
(transfuse-urgently patient7 hospital3 A neg A pos )
(accept donor2 )
(donate donor2 O neg center1 )
(moveto center1 hospital2 O neg patient3 )
(transfuse-urgently patient4 hospital2 O neg O neg )
(accept donor6 )
(donate donor6 O pos center3 )
(moveto center3 hospital1 O pos patient1 )
(transfuse-urgently patient1 hospital1 O pos O pos )
(moveto center3 hospital3 A neg patient5 )
 (transfuse patient5 hospital3 A neg A neg )
 (transfuse patient6 hospital3 AB neg AB neg )
 (transfuse patient3 hospital2 B neg B neg )
 (transfuse patient2 hospital1 B neg AB neg )
 (accept donor4 )
 (donate donor4 B pos center2 )
 (accept donor7 )
 (donate donor7 AB neg center3 )
 (accept donor3 )
 (donate donor3 A pos center1 )
 (reject donor8 )
 (reject donor1 )
 (reject donor5 )
```

opt-hmax

```
(accept donor6 )
(donate donor6 O pos center3 )
(moveto center3 hospital1 O pos patient1 )
(transfuse-urgently patient1 hospital1 O pos O pos )
(transfuse-urgently patient7 hospital3 A neg A pos )
(accept donor2 )
(donate donor2 O neg center1 )
(moveto center1 hospital2 O neg patient4 )
(transfuse-urgently patient4 hospital2 O neg O neg )
(moveto center3 hospital3 A neg patient6 )
 (accept donor3 )
 (accept donor4 )
 (accept donor7 )
 (reject donor1 )
 (reject donor8 )
 (reject donor5 )
 (donate donor3 A pos center1 )
 (donate donor7 AB neg center3 )
 (donate donor4 B pos center2 )
 (transfuse patient3 hospital2 B neg B neg )
 (transfuse patient6 hospital3 AB neg AB neg )
 (transfuse patient2 hospital1 B neg AB neg )
 (transfuse patient5 hospital3 A neg A neg )
```

Problem 6 - More Advanced Scenarios



IndiGolog Reasoning Tasks

IndiGolog

High-level logic-based programming language that allows modeling of dynamic environments, agent behavior, and unexpected events

Main modifications

- Simplification of domain and problem to manage the complexity of IndiGolog reasoning tasks
- Single initial situation, inspired by settings of **Problem3**
- Urgent cases are managed through a **Basic Controller**
- Three reasoning tasks
  - **Legality Task**
  - **Projection Task**
  - **Reactive Controller**

IndiGolog Domain - Predicates & Fluents

Predicates

```
% Donors, patients, centers e hospitals
donor(donor1). donor(donor2). donor(donor3). donor(donor4).
patient(patient1). patient(patient2). patient(patient3). patient(patient4).
center(center1). center(center2).
hospital(hospital1). hospital(hospital2).
bloodgroup(a). bloodgroup(b). bloodgroup(ab). bloodgroup(o).
rhf(pos). rhf(neg).

% Compatibility (modeled as predicates)
abo_compatible(a, a). abo_compatible(a, o).
abo_compatible(b, b). abo_compatible(b, o).
abo_compatible(ab, a). abo_compatible(ab, b).
abo_compatible(ab, ab). abo_compatible(ab, o). abo_compatible(o, o).
rh_compatible(pos, pos). rh_compatible(neg, neg). rh_compatible(pos, neg).

% Blood groups
has_bloodgroup(donor1, a). has_bloodgroup(donor2, o).
has_bloodgroup(donor3, o). has_bloodgroup(donor4, o).
has_bloodgroup(patient1, b). has_bloodgroup(patient2, a).
has_bloodgroup(patient3, b). has_bloodgroup(patient4, o).

% Rh factor
has_rhf(donor1, pos). has_rhf(donor2, neg).
has_rhf(donor3, neg). has_rhf(donor4, pos).
has_rhf(patient1, neg). has_rhf(patient2, pos).
has_rhf(patient3, pos). has_rhf(patient4, neg).
```

Relational Fluents

```
% Relational fluents
rel_fluent(donor_at(D, C)) :- donor(D), center(C).
rel_fluent(patient_at(P, H)) :- patient(P), hospital(H).
rel_fluent(eligible(D)) :- donor(D).
rel_fluent(donated(D)) :- donor(D).
rel_fluent(checked(D)) :- donor(D).
rel_fluent(needs_transfusion(P)) :- patient(P).
rel_fluent(urgent(P)) :- patient(P).
rel_fluent(some_exo).
```

Functional Fluents

```
% Functional fluents
fun_fluent(available_bags(BG, RH, L)) :- bloodgroup(BG), rhf(RH), (center(L) ; hospital(L)).
fun_fluent(donor_age(D)) :- donor(D).
fun_fluent(hemoglobin(D)) :- donor(D).
fun_fluent(max_pressure(D)) :- donor(D).
fun_fluent(min_pressure(D)) :- donor(D).
```

IndiGolog Domain - Initial Situation

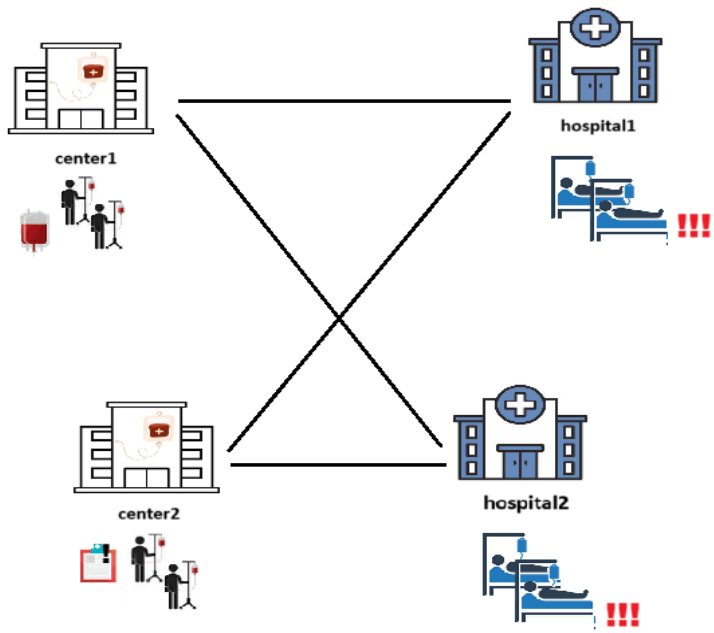
```
initially(donor_at(donor1, center1), true).
initially(donor_at(donor2, center1), true).
initially(donor_at(donor3, center2), true).
initially(donor_at(donor4, center2), true).

initially(donor_age(donor1), 30).
initially(hemoglobin(donor1), 13.0).
initially(max_pressure(donor1), 120).
...
```

```
initially(patient_at(patient1, hospital1), true).
initially(patient_at(patient2, hospital1), true).
initially(patient_at(patient3, hospital2), true).
initially(patient_at(patient4, hospital2), true).
```

```
initially(urgent(patient1), false).
initially(urgent(patient2), true).
initially(urgent(patient3), false).
initially(urgent(patient4), true).
```

```
initially(available_bags(a, pos, center1), 0).
initially(available_bags(a, neg, center1), 0).
initially(available_bags(b, pos, center1), 0).
initially(available_bags(b, neg, center1), 0).
...
```




```
% Exogeneous actions flag
initially(some_exo, false).
```

```
initially(needs_transfusion(P), true) :- patient(P), member(P, [patient1, patient2, patient3, patient4]).
initially(needs_transfusion(P), false) :- patient(P), \+ initially(patient(P), true).
```

. . .

IndiGolog Domain - Primitive Actions

Primitive actions executable in **all tasks**

- `check(D)` Evaluates donor eligibility.
- `donate(D, BG, RH, C)` Performs donation and updates center stock.
- `moveto(C, H, BG, RH, P)` Transports blood bags between centers and hospitals.
- `transfuse(P, H, BG, RH, PBG, PRH)` Executes a transfusion if compatible blood is available.

Primitive actions used as **helpers** for the basic and reactive controllers

- `skip_patient` and `already_transfused` for handling urgent transfusions.
- `reject_donor` for reject donors.

IndiGolog Domain - Exogenous Actions

External, non-deterministic events which affect the system's evolution

emergency(P) A previously stable patient now needs a urgent transfusion

```
exog_action(emergency(P)) :- patient(P).
poss(emergency(P), patient(P)).
causes_true(emergency(P), urgent(P), true).
causes_true(emergency(P), needs_transfusion(P), true).
```

unavailable(D) After checks, a donor becomes unavailable for donating.

```
exog_action(unavailable(D)) :- donor(D).
poss(unavailable(D), donor(D)).
causes_false(unavailable(D), eligible(D), true).
```

change_pressure(D, NewMax, NewMin) After checks, the pressure of a donor suddenly change

```
exog_action(change_pressure(D, _, _)) :- donor(D).
poss(change_pressure(D, _, _), donor(D)).
causes_val(change_pressure(D, NewMax, _), max_pressure(D), N, N is NewMax).
causes_val(change_pressure(D, _, NewMin), min_pressure(D), N, N is NewMin).
```

Task 1 - Legality Task

Legality Task

Verify whether a given **sequence of actions** is executable from the initial state, checking that complex operations such as donation, transport and transfusion follow domain constraints.

$$\mathcal{D} \models executable(do([\alpha_1, \ldots, \alpha_n], S_0))$$

Sequence of actions

accept(donor1);

donate(donor1 A pos center1);

moveto(center1 hospital2 A pos patient2);

transfuse(patient2 hospital2 A pos A pos);

```
proc(actions_legality, [check(donor1),
donate(donor1, a, pos, center1),
moveto(center1, hospital1, a, pos,
patient2), transfuse(patient2,
hospital1, a, pos, a, pos)]).

legality :- indigolog(actions_legality).
```

PROGRAM: Program has executed to completion!! History done:

[transfuse(patient2,hospital1,a,pos,a,pos),moveto(center1,hospital1,

a,pos,patient2),donate(donor1,a,pos,center1),check(donor1)]

Task 2 - Projection Task

Projection Task

Assess whether a fluent (i.e., a property of the world) holds in the resulting situation after performing a specific sequence of actions.

$$\mathcal{D} \models Q(do([\alpha_1, \ldots, \alpha_n], S_0)).$$

Query: patient successfully received transfusion

```
proc(actions_seq, [check(donor1), donate(donor1, a, pos, center1),
moveto(center1, hospital1, a, pos, patient2), transfuse(patient2,
hospital1, a, pos, a, pos)]).
proc(fcond, ?(neg(needs_transfusion(patient2)))).

projection :- indigolog([actions_seq, fcond]).
```

(not (needs-transfusion(patient2))

PROGRAM: Program has executed to completion!! History done:

[transfuse(patient2,hospital1,a,pos,a,pos),moveto(center1,hospital1,

a,pos,patient2),donate(donor1,a,pos,center1),check(donor1)] **true.**

Controllers - Basic Controller

Basic Controller

Sequential controller that executes all core tasks, implemented as procedures

Donation Procedures control the process of checking donors and performing donations

* `check_all` : Recursively checks eligibility for all donors.
* `donate_all` : Recursively performs donations for all eligible donors.
* Ineligible donors trigger the fallback primitive action `reject_donor(D)` .

Transfusion Procedures handle the movement of blood and transfusions to patients, prioritizing urgent cases

* `transfuse_all_urgent` : Prioritizes transfusion for urgent patients
* `transfuse_all` : Performs transfusion for all patients who still need it
* Flow controlled through `skip_patient(P)` and `already_transfused(P)` .

```
proc(control(basic), [check_all, donate_all, transfuse_all_urgent, transfuse_all]).
```

Controllers - Reactive Controller

Reactive Controller

Non-deterministic, interrupt-driven controller that responds **dynamically** to **exogenous events**

The controller is able to **interrupts** the current plan and **re-plans online**

* Uses *prioritized_interrupts* to pause the basic controller when `some_exo` becomes true.
* Invokes the `handle_event` procedure, which checks and handles exogenous events.

```
proc(handle_event,
    [
        if(some_exo,
            nondet(
                exists(P, if(emergency(P), handle_emergency(P), [])),
                exists(D, if(unavailable(D), handle_unavailable(D), [])),
                exists(D, exists(NewMax, exists(NewMin,
                    if(change_pressure(D, NewMax, NewMin), handle_pressure_change(D, NewMax, Ne
                )))
            ),
            []
        )
    ]).
```

```
proc(control(reactive), [
    prioritized_interrupts([
        interrupt(some_exo, [
            if(some_exo, unset(some_exo), []),
            gexec(neg(some_exo), handle_event)
        ])
    ]),
    control(basic)
]).
```

Controllers - Output Plans

Basic Controller

[already_transfused(patient4),transfuse(patient3,hospital2,o,pos,b,pos),moveto(center2,hospital2,o,pos,patient3),already_transfused(patient2),transfuse(patient1,hospital1,o,neg,b,neg),moveto(center1,hospital1,o,neg,patient1),transfuse(patient4,hospital2,o,neg,o,neg),moveto(center1,hospital2,o,neg,patient4),skip_patient(patient3),transfuse(patient2,hospital1,a,pos,a,pos),moveto(center1,hospital1,a,pos,patient2),skip_patient(patient1),donate(donor4,o,pos,center2),donate(donor3,o,neg,center2),reject_donor(donor2),donate(donor1,a,pos,center1),check(donor4),check(donor3),check(donor2),check(donor1)]

Reactive Controller

[already_transfused(patient4),already_transfused(patient3),already_transfused(patient2),transfuse(patient1,hospital1,o,neg,b,neg),moveto(center2,hospital1,o,neg,patient1),transfuse(patient4,hospital2,o,neg,o,neg),moveto(center1,hospital2,o,neg,patient4),**transfuse(patient3,hospital2,o,neg,b,pos)**,moveto(center1,hospital2,o,neg,patient3),transfuse(patient2,hospital1,a,pos,a,pos),moveto(center1,hospital1,a,pos,patient2),**emergency(patient3)**,skip_patient(patient1),**reject_donor(donor4)**,donate(donor3,o,neg,center2),reject_donor(donor2),**unavailable(donor4)**,donate(donor1,a,pos,center1),check(donor4),check(donor3),check(donor2),check(donor1),stop_interrupts]

unavailable(donor4) emergency(patient3)

Thank you for listening!

Blood Transfusion Service via PDDL and IndiGolog

Planning & Reasoning Project