Tugas Besar
Matematika Diskrit FOL

Kompilasi FOL dan PySwip

032 – 061

Reida – Shofiana

DAFTAR ISI

BAB I
PENDAHULUAN

I.1 Analisis Isu

Untuk memetakan penyebab-penyebab dari isu terbatasnya ketersediaan ruang kelas yang dapat digunakan secara bersamaan selama masa ujian, digunakan Diagram Fishbone (Tulang Ikan) dengan rincian diagram sebagai berikut:

A. Kepala Ikan (Masalah Utama):
   Bentroknya Jadwal Ujian
B. Tulang Ikan (Kategori Penyebab):
   1. Ruangan
   2. Mata Kuliah
   3. Jumlah Kelas
   4. Pengawas
C. Duri-duri kecil (Rincian Penyebab Spesifik):
   1. Ruangan
      - Jumlah ruangan di gedung JTK tidak diperhitunkan.
      - Jam saat ruangan sedang digunakan ada yang sama.
   2. Mata Kuliah
      - Tidak memperhatikan mata kuliah dari setiap tingkat.
      - Perbedaan jenis mata kuliah (teori atau praktek).
   3. Jumlah Kelas
      - Tidak memperhatikan jumlah kelas dari setiap tingkat dan prodi.
   4. Pengawas
      - Pengawas dapat hadir atau tidak pada jadwal.
      - Pengawas yang tersedia mengawasi lebih dari satu kelas di saat yang sama.

Gambar 1 Diagram Fishbone

I.2 Pernyataan Masalah

Dari hasil analisis Fishbone, dapat dirumuskan sebuah pertanyaan spesifik untuk diselesaikan dengan FOL, yaitu:

“Bagaimana menyusun penjadwalan waktu ETS agar semua mata kuliah di setiap program studi dan tingkat dapat terujikan, tanpa ada terjadinya bentrok antara ruangan yang digunakan, kelas yang menjalani ujian, dan pengawas yang ditugaskan?”

BAB II
PEMODELAN LOGIKA ORDE PERTAMA

II.1 Formulasi FOL

Pemodelan sistem penjadwalan ujian dilakukan menggunakan Logika Orde Pertama (First Order Logic / FOL) yang direpresentasikan dalam bentuk predikat, fakta, rules, dan rantai inferensi tiga langkah. Model ini digunakan untuk menganalisis potensi konflik penjadwalan, ketersediaan ruangan, dan kesiapan pengawas.

II.1.1 Predikat

Berikut adalah predikat utama yang digunakan dalam sistem, dengan aritas ≥ 1:

1) jumlah_ruang(Gedung, Jumlah)
2) ketersediaan_ruang(Ruang, Status)
3) jadwal_penggunaan(Ruang, Waktu, Status)
4) mata_kuliah(MK, Tingkat, Tipe)
5) jumlah_kelas(Prodi, Tingkat, Jumlah)
6) kelas_dijadwalkan(Kelas, MK, Waktu)
7) kelas_di_ruang(Kelas, Ruang, Waktu)
8) pengawas(Pengawas, Status)
9) ditugaskan(Pengawas, Kelas, Waktu)
10) kebutuhan_ruang(TipeMK, JenisRuang)

II.1.2 Fakta

Kondisi nyata sistem penjadwalan ujian pada suatu periode tertentu direpresentasikan menjadi fakta-fakta berikut:

1) jumlah_ruang(gedung_jtk, 6).
2) ketersediaan_ruang(ruang_201, terpakai).
3) ketersediaan_ruang(ruang_301, tersedia).
4) ketersediaan_ruang(ruang_401, tersedia).
5) jadwal_penggunaan(ruang_201, jam_9, terpakai).
6) jadwal_penggunaan(ruang_301, jam_9, tersedia).
7) jadwal_penggunaan(ruang_401, jam_14, tersedia).
8) mata_kuliah(mk_algoritma, tingkat_1, teori).
9) mata_kuliah(mk_jarkom, tingkat_2, teori).
10) mata_kuliah(mk_basisdata, tingkat_1, teori).
11) jumlah_kelas(tif, tingkat_1, 7).
12) jumlah_kelas(tko, tingkat_1, 2).
13) pengawas(pengawas_andi, hadir).
14) pengawas(pengawas_budi, tidak_hadir).
15) pengawas(pengawas_citra, hadir).
16) kelas_dijadwalkan(kelasA, mk_algoritma, jam_9).

17) kelas_dijadwalkan(kelasB, mk_jarkom, jam_9).
18) kelas_dijadwalkan(kelasC, mk_basisdata, jam_14).
19) butuh_ruang(kelasA).
20) butuh_ruang(kelasB).
21) butuh_ruang(kelasC).
22) kelas_di_ruang(kelasA, ruang_201, jam_9).
23) kelas_di_ruang(kelasB, ruang_201, jam_9).
24) kelas_di_ruang(kelasC, ruang_401, jam_14).
25) ditugaskan(pengawas_andi, kelasA, jam_9).
26) ditugaskan(pengawas_budi, kelasB, jam_9).
27) ditugaskan(pengawas_citra, kelasC, jam_14).
28) kebutuhan_ruang(teori, ruang_kelas).
29) kebutuhan_ruang(praktek, lab).

II.1.3 Rules

Rules dalam notasi FOL menggunakan quantifier universal (∀) dan implikasi (→) dituliskan sebagai berikut:

1) Rule 1 : Bentrok Ruangan

$$\forall K_1 \forall K_2 \forall R \forall W \ (kelas\_di\_ruang(K_1, R, W) \wedge kelas\_di\_ruang(K_2, R, W) \wedge K_1 \neq K_2) \rightarrow bentrok\_ruangan(K_1, K_2)$$

2) Rule 2 : Ruangan Tidak Dapat Digunakan

$$\forall R \forall W \ (jadwal\_penggunaan(R, W, terpakai) \wedge ketersediaan\_ruang(R, terpakai)) \rightarrow ruang\_tidak\_dapat\_dipakai(R, W)$$

3) Rule 3 : Kekurangan Ruang

$$\forall P \forall T \forall JK \forall JR \ (jumlah\_kelas(P, T, JK) \wedge jumlah\_ruang(gedung\_jtk, JR) \wedge JK > JR) \rightarrow kekurangan\_ruang(P, T)$$

4) Rule 4 : Potensi Bentrok Ruang

$$\forall K \forall R \forall W \ (butuh\_ruang(K) \wedge kelas\_di\_ruang(K, R, W) \wedge ruang\_tidak\_dapat\_dipakai(R, W)) \rightarrow potensi\_bentrok(K, R, W)$$

5) Rule 5 : Pengawas Tidak Bisa Mengawasi

$$\forall P \forall K \forall W \ (ditugaskan(P, K, W) \wedge pengawas(P, tidak\_hadir)) \rightarrow pengawas\_tidak\_bisa(P, K)$$

6) Rule 6 : Kelas Bermasalah

$$\forall K \ (bentrok\_ruangan(K, X) \vee potensi\_bentrok(K, R, W)) \rightarrow kelas\_bermasalah(K)$$

7) Rule 7 : Jadwal Bermasalah

$$\forall K\ (kelas\_bermasalah(K) \wedge kelas\_dijadwalkan(K, M, W)) \rightarrow jadwal\_bermasalah(K)$$

8) Rule 8 : Perlu Penjadwalan Ulang

$$\forall K\ jadwal\_bermasalah(K) \rightarrow perlu\_penjadwalan\_ulang(K)$$

9) Rule 9 : Butuh Pengawas Pengganti

$$\forall K\ pengawas\_tidak\_bisa(P, K) \rightarrow butuh\_pengawas\_pengganti(K)$$

II.1.4 Rantai 3 Langkah

Sistem memiliki rantai inferensi tiga langkah yang digunakan untuk menyimpulkan bahwa suatu kelas perlu dijadwalkan ulang, yaitu:

$$bentrok\_ruangan(K, X) \rightarrow kelas\_bermasalah(K) \rightarrow jadwal\_bermasalah(K) \rightarrow perlu\_penjadwalan\_ulang(K)$$

Dalam notasi FOL formal:

$$\forall K\ ((bentrok\_ruangan(K, X) \rightarrow kelas\_bermasalah(K)) \wedge (kelas\_bermasalah(K) \rightarrow jadwal\_bermasalah(K)) \wedge (jadwal\_bermasalah(K) \rightarrow perlu\_penjadwalan\_ulang(K)))$$

Rantai ini menunjukkan bahwa jika suatu kelas mengalami bentrok ruangan, maka kelas tersebut bermasalah, menyebabkan jadwal ujian bermasalah, dan akhirnya memerlukan penjadwalan ulang.

II.2 Struktur Knowledge Base

```
% ====================================================================
% FILE: prolog_kb.pl
% Knowledge Base Logika Orde Pertama (FOL)
% TEMA: Bentrok Penjadwalan Ujian ETS
% ====================================================================
% Predikat:
% jumlah_ruang/2, ketersediaan_ruang/2, mata_kuliah/3,
% jumlah_kelas/3, jadwal_penggunaan/3, pengawas/2,
% kelas_dijadwalkan/3, butuh_ruang/1,
% kelas_di_ruang/3, ditugaskan/3, kebutuhan_ruang/2
% ====================================================================

% --------------------------------------------------------------------
% FACTS
% --------------------------------------------------------------------
```

```
% Kapasitas ruang di gedung
jumlah_ruang(gedung_jtk, 6).

% Status ruang (umum)
ketersediaan_ruang(ruang_201, terpakai).
ketersediaan_ruang(ruang_301, tersedia).
ketersediaan_ruang(ruang_401, tersedia).

% Jadwal penggunaan ruang (slot waktu)
jadwal_penggunaan(ruang_201, jam_9, terpakai).
jadwal_penggunaan(ruang_301, jam_9, tersedia).
jadwal_penggunaan(ruang_401, jam_14, tersedia).

% Mata kuliah
mata_kuliah(mk_algoritma, tingkat_1, teori).
mata_kuliah(mk_jarkom, tingkat_2, teori).
mata_kuliah(mk_basisdata, tingkat_1, teori).

% Jumlah kelas per prodi-tingkat
jumlah_kelas(tif, tingkat_1, 7).
jumlah_kelas(tko, tingkat_1, 2).

% Pengawas dan kehadiran
pengawas(pengawas_andi, hadir).
pengawas(pengawas_budi, tidak_hadir).
pengawas(pengawas_citra, hadir).

% Kelas dijadwalkan (kelas, mk, waktu)
kelas_dijadwalkan(kelasA, mk_algoritma, jam_9).
kelas_dijadwalkan(kelasB, mk_jarkom, jam_9).
kelas_dijadwalkan(kelasC, mk_basisdata, jam_14).

% Kelas butuh ruang (semua ujian butuh ruang)
butuh_ruang(kelasA).
butuh_ruang(kelasB).
butuh_ruang(kelasC).

% Penempatan kelas ke ruang (kelas, ruang, waktu)
kelas_di_ruang(kelasA, ruang_201, jam_9).
kelas_di_ruang(kelasB, ruang_201, jam_9).   % sengaja dibuat konflik ruang yang
nyata
kelas_di_ruang(kelasC, ruang_401, jam_14).

% Penugasan pengawas (pengawas, kelas, waktu)
ditugaskan(pengawas_andi, kelasA, jam_9).
ditugaskan(pengawas_budi, kelasB, jam_9).   % budi tidak hadir → masalah
spesifik ke kelasB
ditugaskan(pengawas_citra, kelasC, jam_14).

% Kebutuhan ruang berdasarkan tipe MK (contoh sederhana)
```

```
kebutuhan_ruang(teori, ruang_kelas).
kebutuhan_ruang(praktek, lab).


% ----------------------------------------------------------------
% RULES
% ----------------------------------------------------------------

% Rule 1 — Bentrok Ruangan yang benar-benar “ruang sama, waktu sama”
bentrok_ruangan(K1, K2) :-
    kelas_di_ruang(K1, R, W),
    kelas_di_ruang(K2, R, W),
    K1 @< K2.

% Rule 2 — Ruangan tidak dapat dipakai jika pada waktu itu terpakai + statusnya
terpakai
ruang_tidak_dapat_dipakai(R, W) :-
    jadwal_penggunaan(R, W, terpakai),
    ketersediaan_ruang(R, terpakai).

% Rule 3 — Kekurangan ruang jika jumlah kelas prodi-tingkat melebihi jumlah
ruang gedung
kekurangan_ruang(Prodi, Tingkat) :-
    jumlah_kelas(Prodi, Tingkat, JK),
    jumlah_ruang(gedung_jtk, JR),
    JK > JR.

% Rule 4 — Potensi bentrok: kelas butuh ruang pada W, tapi ruang yang dipakai
pada W ternyata tidak bisa dipakai
% (mengaitkan butuh_ruang + kelas_di_ruang + ruang_tidak_dapat_dipakai)
potensi_bentrok(K, R, W) :-
    butuh_ruang(K),
    kelas_di_ruang(K, R, W),
    ruang_tidak_dapat_dipakai(R, W).

% Rule 5 — Pengawas tidak bisa mengawasi kelas tertentu jika (ditugaskan ke
kelas tsb) dan dia tidak hadir
% Ini memperbaiki “budi otomatis tidak bisa untuk semua kelas”.
pengawas_tidak_bisa(P, K) :-
    ditugaskan(P, K, _W),
    pengawas(P, tidak_hadir).

% Rule 6 — Kelas bermasalah jika bentrok ruangan ATAU terkena potensi bentrok
(ruang tidak bisa dipakai)
kelas_bermasalah(K) :-
    bentrok_ruangan(K, _).
kelas_bermasalah(K) :-
    bentrok_ruangan(_, K).
kelas_bermasalah(K) :-
    potensi_bentrok(K, _, _).
```

```
% Rule 7 – Jadwal bermasalah jika kelas bermasalah dan kelas itu memang
dijadwalkan ujian
jadwal_bermasalah(K) :-
    kelas_bermasalah(K),
    kelas_dijadwalkan(K, _, _).

% Rule 8 – Perlu penjadwalan ulang jika jadwal bermasalah
perlu_penjadwalan_ulang(K) :-
    jadwal_bermasalah(K).

% Rule 9 – Perlu pengawas pengganti jika pengawas tidak bisa mengawasi kelas
tersebut
butuh_pengawas_pengganti(K) :-
    pengawas_tidak_bisa(_, K).
```

II.3 Mekanisme Unifikasi

Unifikasi adalah proses pencocokan dua ekspresi logika dengan cara menemukan Most General Unifier (MGU), yaitu substitusi variabel paling umum yang membuat dua predikat menjadi identik. Dalam Prolog, unifikasi digunakan untuk mencocokkan goal (query) dengan head rule serta mengikat variabel selama proses inferensi.

II.3.1 Unifikasi pada Rule Kompleks (2 Langkah)

Rule kompleks yang dianalisis adalah Rule 4 : Potensi Bentrok Ruang, karena melibatkan lebih dari satu predikat pada bagian badan (body) dan membentuk inferensi berantai.

potensi_bentrok(K, R, W) :-

butuh_ruang(K),

kelas_di_ruang(K, R, W),

ruang_tidak_dapat_dipakai(R, W).

Rule ini memerlukan dua langkah unifikasi utama, yaitu unifikasi antara butuh_ruang(K) dan fakta kebutuhan ruang kelas, serta unifikasi lanjutan antara kelas_di_ruang(K, R, W) dan ruang_tidak_dapat_dipakai(R, W).

II.3.2 Tahap MGU (1 Langkah Unifikasi)

Untuk menjawab query potensi_bentrok(kelasA, R, W), maka langkahnya sebagai berikut:

1) Langkah Unifikasi Pertama
   Pertama substitusi untuk butuh_ruang(K)
   MGU tahap pertama:

$$\theta_1 = \{K \leftarrow kelasA\}$$

2) Langkah Unifikasi Kedua
Setelah substitusi, predikat berikutnya menjadi kelas_di_ruang(kelasA, R, W).
MGU tahap kedua:

$$\theta_2 = \{R \leftarrow ruang\_201, W \leftarrow jam\_9\}$$

3) Tabel MGU

Elemen	Predikat pada Rule	Fakta di KB	Substitusi
1	butuh_ruang(K)	butuh_ruang(kelasA)	{ K / kelasA }
2	kelas_di_ruang(kelasA, R, W)	kelas_di_ruang(kelasA, ruang_201, jam_9)	{ R / ruang_201, W / jam_9 }
3	ruang_tidak_dapat_dipakai(R,W)	ruang_tidak_dapat_dipakai(ruang201, jam_9)	{ R / ruang_201, W / jam_9 }

II.3.3 MGU Akhir dan Validasi Rule

MGU gabungan:

$$\theta = \{K \leftarrow kelasA, R \leftarrow ruang\_201, W \leftarrow jam\_9\}$$

Dengan substitusi ini, predikat terakhir:

ruang_tidak_dapat_dipakai(ruang_201, jam_9)

dapat dibuktikan menggunakan Rule 2, karena:

- jadwal_penggunaan(ruang_201, jam_9, terpakai)
- ketersediaan_ruang(ruang_201, terpakai)

Sehingga Prolog berhasil menyimpulkan:

potensi_bentrok(kelasA, ruang_201, jam_9)

II.4 Teori Resolusi

Teori resolusi digunakan untuk membuktikan validitas suatu kesimpulan dengan cara mengonversi rule ke dalam bentuk Conjunctive Normal Form (CNF) dan menunjukkan bahwa penyangkalan terhadap tujuan menghasilkan klausa kosong (□).

II.4.1 Rule yang Dianalisis

Rule yang dianalisis adalah Rule 8 : Perlu Penjadwalan Ulang, karena merupakan kesimpulan akhir dari rantai inferensi sistem.

perlu_penjadwalan_ulang(K) :-

jadwal_bermasalah(K).

II.4.2 Konversi Rule ke CNF

1) Bentuk FOL

$$\forall K\ jadwal\_bermasalah(K) \rightarrow perlu\_penjadwalan\_ulang(K)$$

2) Eliminasi Implikasi

$$\forall K\ (\neg jadwal\_bermasalah(K) \vee perlu\_penjadwalan\_ulang(K))$$

3) Clausal Form

$$\{\neg jadwal\_bermasalah(K) \vee perlu\_penjadwalan\_ulang(K)\}$$

II.4.3 Langkah Pembuktian Resolusi Awal

Untuk membuktikan perlu_penjadwalan_ulang(kelasA), maka:

1) Negasi Tujuan

$$\neg perlu\_penjadwalan\_ulang(kelasA)$$

2) Himpunan Klausa
   - Klausa hasil CNF Rule 8:
     $$\neg jadwal\_bermasalah(K) \vee perlu\_penjadwalan\_ulang(K)$$
   - Fakta hasil inferensi sebelumnya:
     $$jadwal\_bermasalah(kelasA)$$
   - Negasi tujuan:
     $$\neg perlu\_penjadwalan\_ulang(kelasA)$$

II.4.4 Proses Resolusi

Resolusi antara:

$$\neg jadwal\_bermasalah(kelasA) \vee perlu\_penjadwalan\_ulang(kelasA)$$

dan

$$\neg perlu\_penjadwalan\_ulang(kelasA)$$

menghasilkan:

$$\neg jadwal\_bermasalah(kelasA)$$

Resolusi lanjutan dengan klausa:

$$jadwal\_bermasalah(kelasA)$$

Menghasilkan klausa kosong:

$$\square$$

II.4.5 Kesimpulan Resolusi

Diperolehnya klausa kosong (□) menunjukkan bahwa negasi dari perlu_penjadwalan_ulang(kelasA) tidak konsisten dengan Knowledge Base. Dengan demikian, dapat disimpulkan secara formal bahwa perlu_penjadwalan_ulang(kelasA) terbukti benar secara logis melalui metode resolusi.

BAB III
HASIL INFERENSI DAN SOLUSI

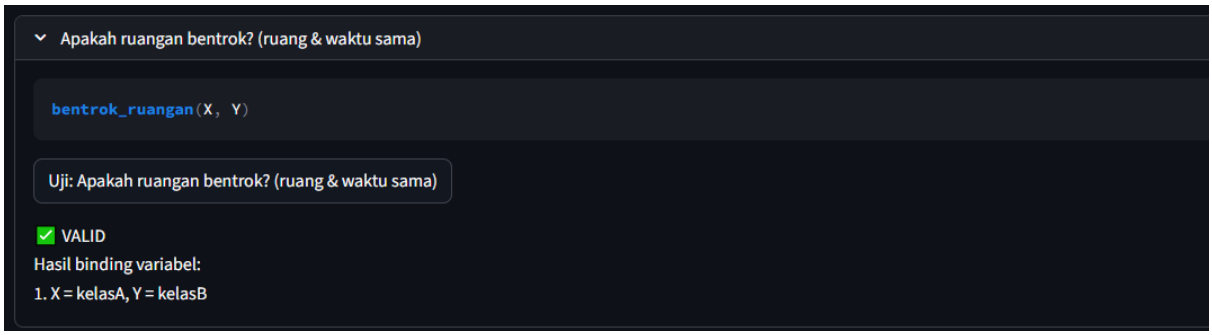
III.1 Hasil Uji Inferensi

A. Query: Apakah ruangan bentrok? (ruang & waktu sama)

```
% Rule 1 – Bentrok Ruangan yang benar-benar "ruang sama, waktu sama"
bentrok_ruangan(K1, K2) :-
    kelas_di_ruang(K1, R, W),
    kelas_di_ruang(K2, R, W),
    K1 @< K2.
```

- kelasA dan kelasB memakai ruang_201 di jam_9
- K1 @< K2 menghilangkan duplikasi simetris

Hasil Uji Inferensi Streamlit:

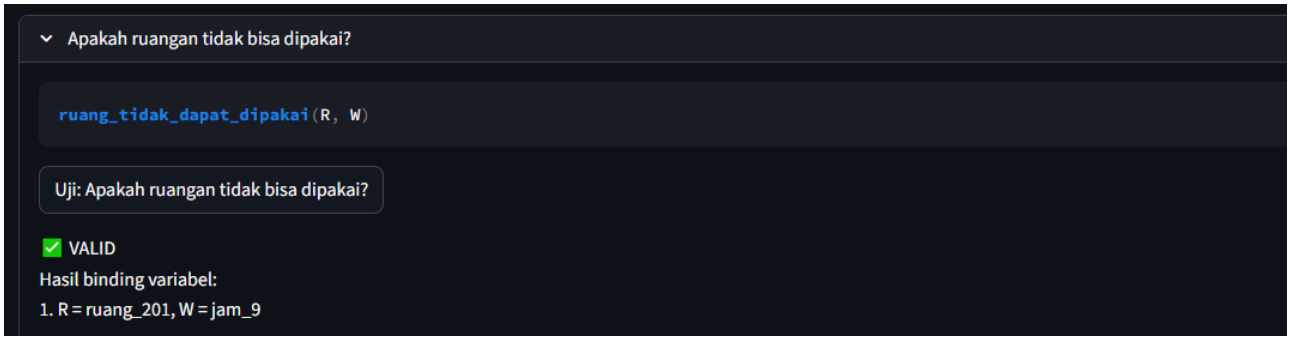


B. Query: Apakah ruangan tidak bisa dipakai?

```
% Rule 2 – Ruangan tidak dapat dipakai jika pada waktu itu terpakai +
statusnya terpakai
ruang_tidak_dapat_dipakai(R, W) :-
    jadwal_penggunaan(R, W, terpakai),
    ketersediaan_ruang(R, terpakai).
```

```
% FACTS
ketersediaan_ruang(ruang_201, terpakai).
jadwal_penggunaan(ruang_201, jam_9, terpakai).
```

Hasil Uji Inferensi Streamlit:

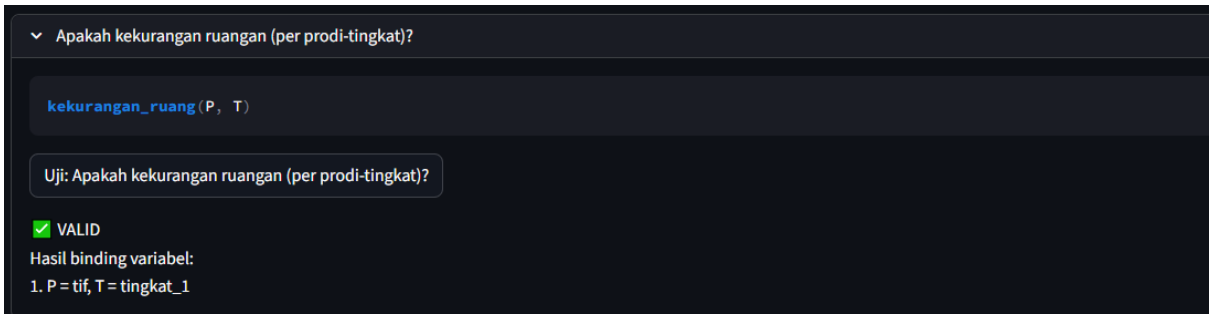


C. Query: Apakah kekurangan ruangan(per prodi-tingkat)?

```
% FACTS

% Kapasitas ruang di gedung
jumlah_ruang(gedung_jtk, 6).

% Jumlah kelas per prodi-tingkat
jumlah_kelas(tif, tingkat_1, 7).
jumlah_kelas(tko, tingkat_1, 2).
```

- Jumlah_kelas tingkat_1 prodi tif ada 7 sedangkan jumlah_ruang hanya ada 6.

Hasil Uji Inferensi Streamlit:



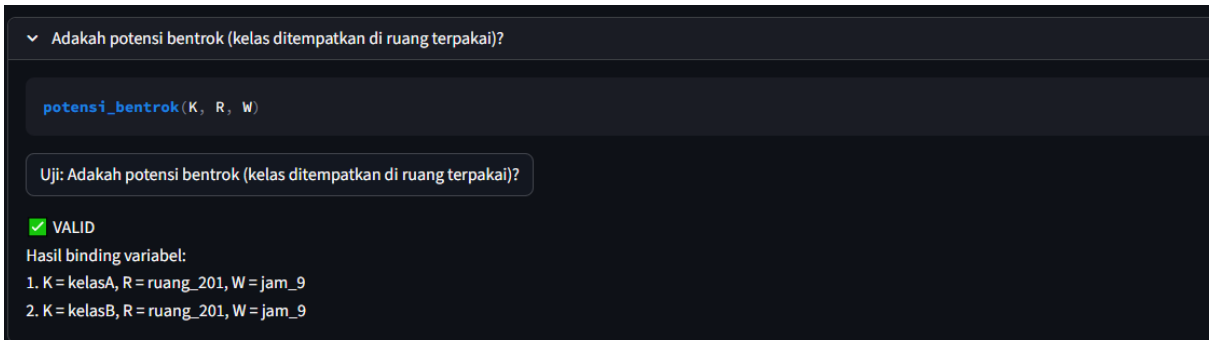
D. Query: Adakah potensi bentrok?

```
% Rule 4 – Potensi bentrok: kelas butuh ruang pada W, tapi ruang yang
dipakai pada W ternyata tidak bisa dipakai
% (mengaitkan butuh_ruang + kelas_di_ruang + ruang_tidak_dapat_dipakai)
potensi_bentrok(K, R, W) :-
    butuh_ruang(K),
    kelas_di_ruang(K, R, W),
    ruang_tidak_dapat_dipakai(R, W).
```

```
% FACTS

% Kelas butuh ruang (semua ujian butuh ruang)
butuh_ruang(kelasA).
butuh_ruang(kelasB).
butuh_ruang(kelasC).

% Penempatan kelas ke ruang (kelas, ruang, waktu)
kelas_di_ruang(kelasA, ruang_201, jam_9).
kelas_di_ruang(kelasB, ruang_201, jam_9).
kelas_di_ruang(kelasC, ruang_401, jam_14).
```

- kelasA dan kelasB butuh ruang, keduanya ditempatkan di ruang_201 pada waktu jam_9.

Hasil Uji Inferensi Streamlit:

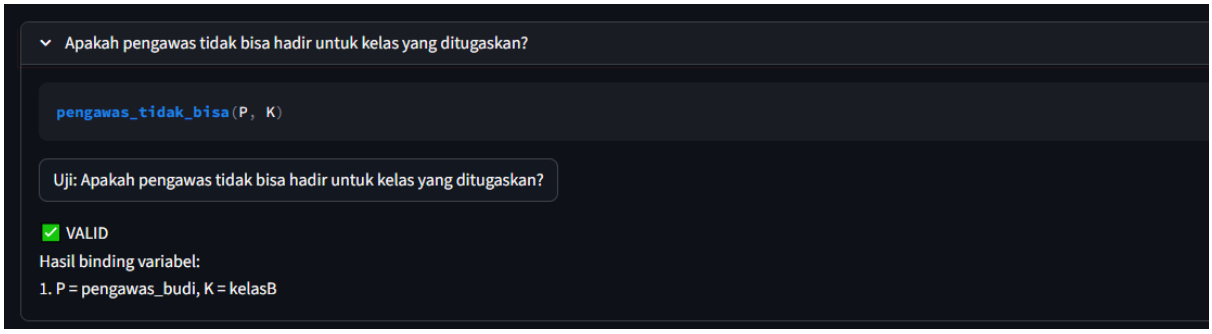


E. Query: Apakah pengawas tidak bisa hadir untuk kelas yang ditugaskan?

```
% FACTS
% Pengawas dan kehadiran
pengawas(pengawas_andi, hadir).
pengawas(pengawas_budi, tidak_hadir).
pengawas(pengawas_citra, hadir).

% Penugasan pengawas (pengawas, kelas, waktu)
ditugaskan(pengawas_andi, kelasA, jam_9).
ditugaskan(pengawas_budi, kelasB, jam_9).
ditugaskan(pengawas_citra, kelasC, jam_14).
```

- Pengawas_budi di kelasB pada waktu jam_9 memiliki status tidak_hadir.

Hasil Uji Inferensi Streamlit:



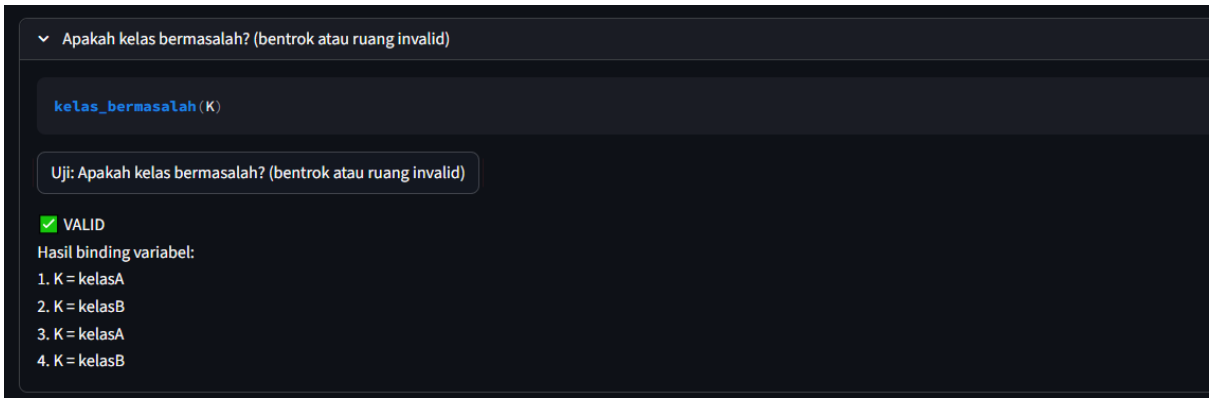
F. Query: Apakah kelas bermasalah? (bentrok atau ruang invalid)

```
% Rule 6 – Kelas bermasalah jika bentrok ruangan ATAU terkena potensi
bentrok (ruang tidak bisa dipakai)
kelas_bermasalah(K) :-
    bentrok_ruangan(K, _).
kelas_bermasalah(K) :-
    bentrok_ruangan(_, K).
kelas_bermasalah(K) :-
    potensi_bentrok(K, _, _).

% Rule 1 – Bentrok Ruangan yang benar-benar “ruang sama, waktu sama”
bentrok_ruangan(K1, K2) :-
    kelas_di_ruang(K1, R, W),
    kelas_di_ruang(K2, R, W),
    K1 @< K2.
```

```
% FACTS
% Penempatan kelas ke ruang (kelas, ruang, waktu)
kelas_di_ruang(kelasA, ruang_201, jam_9).
kelas_di_ruang(kelasB, ruang_201, jam_9).
kelas_di_ruang(kelasC, ruang_401, jam_14).
```

- kelasA → bentrok_ruangan(A,B)
- kelasB → bentrok_ruangan(A,B) dan potensi_bentrok
- kelasC → aman (ruang_401 jam_14 tersedia)

Hasil Uji Inferensi Streamlit:

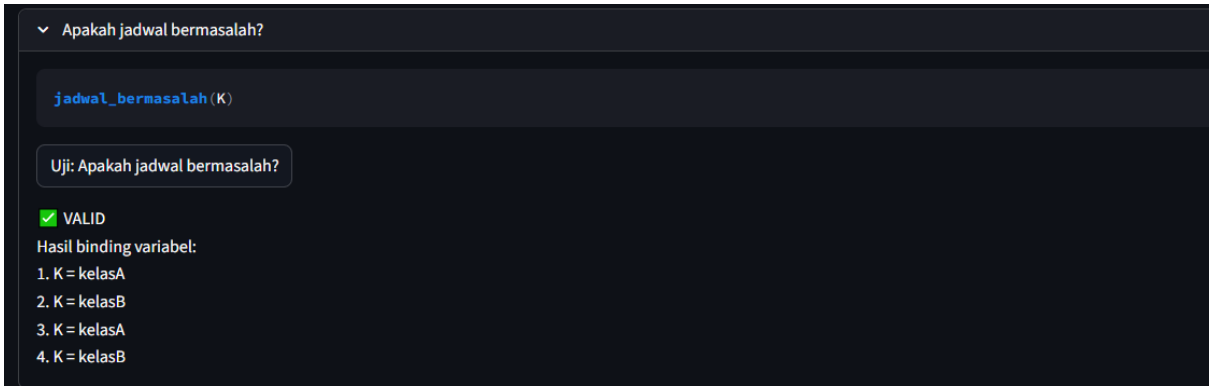


G. Query: Apakah jadwal bermasalah karena bentrok?

```
% Rule 7 – Jadwal bermasalah jika kelas bermasalah dan kelas itu memang
dijadwalkan ujian
jadwal_bermasalah(K) :-
    kelas_bermasalah(K),
    kelas_dijadwalkan(K, _, _).

% Rule 6 – Kelas bermasalah jika bentrok ruangan ATAU terkena potensi
bentrok (ruang tidak bisa dipakai)
kelas_bermasalah(K) :-
    bentrok_ruangan(K, _).
kelas_bermasalah(K) :-
    bentrok_ruangan(_, K).
kelas_bermasalah(K) :-
    potensi_bentrok(K, _, _).

% Rule 1 – Bentrok Ruangan yang benar-benar “ruang sama, waktu sama”
bentrok_ruangan(K1, K2) :-
    kelas_di_ruang(K1, R, W),
    kelas_di_ruang(K2, R, W),
    K1 @< K2.
```

```
% FACTS
% Penempatan kelas ke ruang (kelas, ruang, waktu)
kelas_di_ruang(kelasA, ruang_201, jam_9).
kelas_di_ruang(kelasB, ruang_201, jam_9).
kelas_di_ruang(kelasC, ruang_401, jam_14).
```

Hasil Uji Inferensi Streamlit:

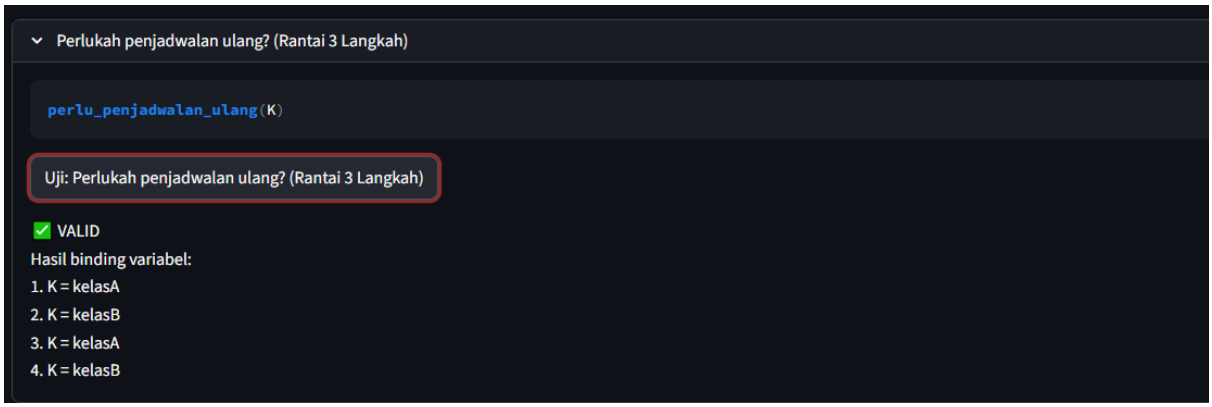


H. Query: Perlukah penjadwalan ulang? (Rantai 3 Langkah)

```
% Rule 8 — Perlu penjadwalan ulang jika jadwal bermasalah
perlu_penjadwalan_ulang(K) :-
    jadwal_bermasalah(K).

% Rule 7 — Jadwal bermasalah jika kelas bermasalah dan kelas itu memang
dijadwalkan ujian
jadwal_bermasalah(K) :-
    kelas_bermasalah(K),
    kelas_dijadwalkan(K, _, _).

% Rule 6 — Kelas bermasalah jika bentrok ruangan ATAU terkena potensi
bentrok (ruang tidak bisa dipakai)
kelas_bermasalah(K) :-
    bentrok_ruangan(K, _).
kelas_bermasalah(K) :-
    bentrok_ruangan(_, K).
kelas_bermasalah(K) :-
    potensi_bentrok(K, _, _).

% Rule 4 — Potensi bentrok: kelas butuh ruang pada W, tapi ruang yang
dipakai pada W ternyata tidak bisa dipakai
% (mengaitkan butuh_ruang + kelas_di_ruang + ruang_tidak_dapat_dipakai)
potensi_bentrok(K, R, W) :-
    butuh_ruang(K),
    kelas_di_ruang(K, R, W),
    ruang_tidak_dapat_dipakai(R, W).

% Rule 1 — Bentrok Ruangan yang benar-benar “ruang sama, waktu sama”
bentrok_ruangan(K1, K2) :-
    kelas_di_ruang(K1, R, W),
    kelas_di_ruang(K2, R, W),
    K1 @< K2.
```

```
% FACTS
% Penempatan kelas ke ruang (kelas, ruang, waktu)
kelas_di_ruang(kelasA, ruang_201, jam_9).
kelas_di_ruang(kelasB, ruang_201, jam_9).
kelas_di_ruang(kelasC, ruang_401, jam_14).
```

Hasil Uji Inferensi Streamlit:

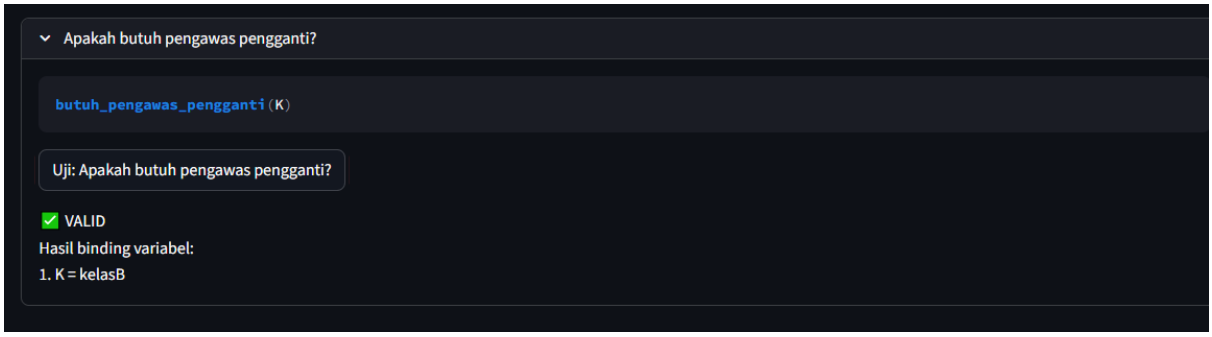


I. Query: Apakah butuh pengawas pengganti?

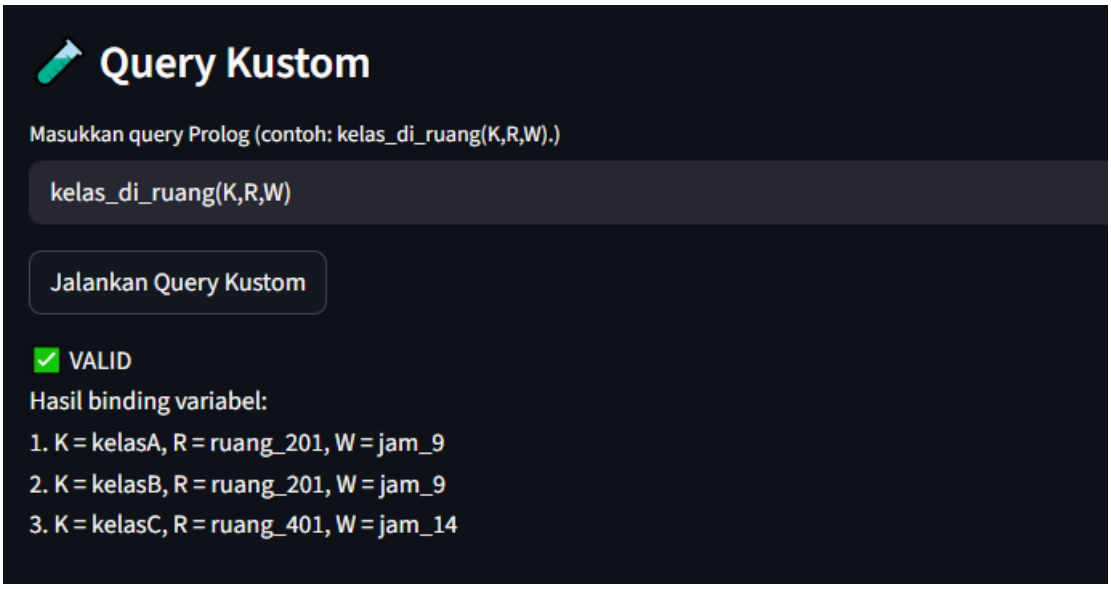
```
% Rule 9 – Perlu pengawas pengganti jika pengawas tidak bisa mengawasi
kelas tersebut
butuh_pengawas_pengganti(K) :-
    pengawas_tidak_bisa(_, K).

% Rule 5 – Pengawas tidak bisa mengawasi kelas tertentu jika (ditugaskan
ke kelas tsb) dan dia tidak hadir
% Ini memperbaiki “budi otomatis tidak bisa untuk semua kelas”.
pengawas_tidak_bisa(P, K) :-
    ditugaskan(P, K, _W),
    pengawas(P, tidak_hadir).
```

```
% FACTS
% Penugasan pengawas (pengawas, kelas, waktu)
ditugaskan(pengawas_andi, kelasA, jam_9).
ditugaskan(pengawas_budi, kelasB, jam_9).
ditugaskan(pengawas_citra, kelasC, jam_14).

% Pengawas dan kehadiran
pengawas(pengawas_andi, hadir).
pengawas(pengawas_budi, tidak_hadir).
pengawas(pengawas_citra, hadir).
```

Hasil Uji Inferensi Streamlit:



J. Query Kustom: kelas_di_ruang(K,R,W)



III.2 Analisis Solusi

III.2.1 Analisis Validitas

Salah satu kesalahan penalaran (*fallacy*) yang berhasil disanggah oleh Knowledge Base (KB) ini adalah generalisasi keliru (hasty generalization) dalam penjadwalan. Anggapan bahwa *seluruh kelas pada waktu yang sama pasti bermasalah dan harus dijadwalkan ulang* ini dianggap sebagai kegagalan sistem secara keseluruhan.

Hasil Inferensi menunjukkan bahwa meskipun terdapat bentrok ruangan antara kelasA dan kelasB yang dijadwalkan pada waktu yang sama yaitu jam_9, tidak semua kelas otomatis dikategorikan bermasalah. Melalui rantai inferensi bertingkat, KB membuktikan bahwa sebuah kelas hanya dianggap bermasalah apabila memenuhi beberapa kondisi tertentu, yaitu adanya bentrok jadwal atau penggunaan ruangan yang tidak valid, serta kelas tersebut memang membutuhkan ruangan. Dengan demikian, KB tidak melakukan inferensi berlebihan, melainkan menyaring kondisi secara logis dan sistematis sebelum menarik kesimpulan bahwa suatu kelas benar-benar memerlukan penjadwalan ulang.

Pendekatan ini menegaskan bahwa keputusan kebijakan (penjadwalan ulang) tidak boleh diambil hanya berdasarkan satu indikator tunggal, melainkan harus melalui rangkaian kondisi yang saling terkait. Dengan demikian, KB secara eksplisit menolak kesimpulan instan yang tidak didukung oleh struktur penalaran yang langkap.

III.2.2 Analisis Kritis

Meskipun KB berbasis Logika Orde Pertama (FOL) dengan PySwip mampu memodelkan hubungan sebab-akibat secara logis dan konsisten, model ini memiliki keterbatasan dalam merepresentasikan kondisi dunia nyata secara utuh. Salah satu keterbatasan utama adalah ketidakmampuan FOL dalam menangani ketidakpastian dan probabilitas. Dalam konteks penjadwalan ujian, kondisi seperti kemungkinan ruangan menjadi

tersedia secara dinamis tidak dapat dimodelkan secara kuantitatif menggunakan FOL karena semua kondisi dinilai mutlak (benar atau salah), padahal nyatanya, suatu bentrok bisa bersifat sementara atau masih dapat ditoleransi.

Selain itu, KB tidak dapat mempertimbangkan faktor manusia, seperti kelelahan pengawas, preferensi dosen, atau toleransi terhadap perubahan jadwal. Semua pengawas yang tidak hadir diperlakukan sama tanpa mempertimbangkan alasan atau tingkat keparahannya.

Keterbatasan lainnya adalah sistem tidak bersifat adaptif. Jika terjadi perubahan data, KB tidak belajar dari pengalaman sebelumnya, melainkan hanya menarik kesimpulan dari fakta yang diberikan saat ini.

Dengan demikian, KB ini sangat kuat untuk validasi logika dan pengujian konsistensi keputusan, tetapi masih perlu dikombinasikan dengan pendekatan lain (misalnya sistem berbasis probabilitas atau kebijakan manusia) agar dapat digunakan secara optimal dalam konteks penjadwalan ujian di dunia nyata.

BAB IV
KESIMPULAN

IV.1 Kesimpulan

Knowledge Base berbasis Logika Orde Pertama (FOL) mampu menghasilkan keputusan penjadwalan ujian yang logis, selektif, dan bebas dari generalisasi keliru. Meskipun terdapat kelas yang dijadwalkan pada waktu yang sama, KB tidak langsung mengklasifikasikan seluruh kelas sebagai bermasalah. Melalui rantai inferensi bertingkat, KB hanya menetapkan suatu kelas sebagai bermasalah apabila memenuhi kombinasi kondisi yang relevan, seperti bentrok ruangan nyata, penggunaan ruang yang tidak valid, serta kebutuhan ruang oleh kelas tersebut. Hal ini menunjukkan bahwa KB berhasil menolak kesimpulan instan dan memastikan bahwa kebijakan penjadwalan ulang diambil secara rasional dan terstruktur.

Namun demikian, meskipun kuat dalam validasi logika dan konsistensi keputusan, KB berbasis FOL memiliki keterbatasan dalam merepresentasikan dinamika dan ketidakpastian dunia nyata. Sistem FOL belum mampu menangani probabilitas, faktor manusia, maupun proses pembelajaran adaptif dari pengalaman sebelumnya.

IV.2 Rekomendasi

1) Kebijakan Penjadwalan Ulang Otomatis untuk Kelas Bermasalah
    a. Dasar Inferensi dari KB
    Dari rantai inferensi:
    bentrok_ruangan(kelasA, kelasB)
    ⟶ kelas_bermasalah(kelasA)
    ⟶ jadwal_bermasalah(kelasA)
    ⟶ perlu_penjadwalan_ulang(kelasA)
    KB menyimpulkan bahwa kelasA dan kelasB mengalami bentrok ruang pada waktu yang sama (ruang_201, jam_9).
    b. Rekomendasi Kebijakan
    Untuk mengatasi masalah bentrok ruang pada waktu yang sama, sistem penjadwalan ujian dapat menerapkan mekanisme penjadwalan ulang otomatis terhadap kelas yang terinferensi sebagai perlu_penjadwalan_ulang(K). Implementasi solusi tersebut adalah dengan cara memindahkan kelas ke slot waktu berbeda atau ruang alternatif yang tersedia. Kebijakan ini mengurangi konflik manual dan mempercepat proses validasi jadwal ujian.
2) Kebijakan Validasi Ruang Sebelum Finalisasi Jadwal
    a. Dasar Inferensi dari KB
    KB menghasilkan potensi_bentrok(kelasA, ruang_201, jam_9) karena:
    - ruang_tidak_dapat_dipakai(ruang_201, jam_9)
    - tetapi tetap digunakan oleh kelasA
    b. Rekomendasi Kebijakan

Sebagai tindakan pencegahan atas potensi bentrok, setiap jadwal ujian harus melalui tahap validasi status ruang sebelum ditetapkan sebagai jadwal final. Implementasi solusi tersebut adalah dengan membuat sistem menolak penempatan kelas ke ruang yang sudah terpakai pada waktu tersebut, dan berstatus tidak_dapat_dipakai. Kebijakan ini dapat menghindari terjadinya jadwal ujian fiktif (terjadwal tapi tidak bisa dilaksanakan).

3) Kebijakan Penugasan Pengawas Cadangan
    a. Dasar Inferensi dari KB
    KB menyimpulkan:
        - pengawas_tidak_bisa(pengawas_budi, kelasB)
        - butuh_pengawas_pengganti(kelasB)

    karena pengawas yang ditugaskan tidak hadir.
    b. Rekomendasi Kebijakan
    Untuk mengatasi masalah ketidakhadiran pengawas ini, setiap kelas ujian wajib memiliki minimal satu pengawas cadangan yang otomatis diaktifkan ketika pengawas utama berstatus tidak_hadir. Implementasi solusi tersebut adalah dengan membuat sistem secara otomatis menandai kelas sebagai butuh_pengawas_pengganti dan mencari pengawas lain dengan status hadir. Kebijakan ini dapat mengurangi risiko ujian tertunda akibat absensi pengawas.