LAPORAN PRAKTIKUM 1

MATA KULIAH KECERDASAN KOMPUTASIONAL

“Logical Agents dan Inference dalam Kecerdasan Komputasional”

Disusun Oleh:

Nama: Wayan Raditya Putra

NRP: 5054241029

Program Studi:Rekayasa Kecerdasan Artifisial

Angkatan: 2024

Dosen Pengampu:

Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom

Imam Mustafa Kamal, S.ST, Ph.D

DEPARTEMEN TEKNIK INFORMATIKA

PROGRAM STUDI REKAYASA KECERDASAN ARTIFISIAL

INSTITUT TEKNOLOGI SEPULUH NOPEMBER (ITS)

SURABAYA

2025

Instalasi dan Import Library

Pada tahap awal praktikum, perlu dipastikan seluruh library yang digunakan sudah tersedia. Oleh karena itu dilakukan proses instalasi paket tambahan serta import fungsi-fungsi yang diperlukan di notebook.

In [213…

```
%pip install ipythonblocks
%pip install qpsolvers
from utils import *
from logic import *
from notebook import psource
```

```
Requirement already satisfied: ipythonblocks in d:\ai_journey\computational_intelligence\ci\lib\site-packages (1.9.1)
Requirement already satisfied: ipython>=4.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ipythonblock
s) (9.5.0)
Requirement already satisfied: notebook>=4.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ipythonbloc
ks) (7.4.5)
Requirement already satisfied: requests>=1.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ipythonbloc
ks) (2.32.5)
Requirement already satisfied: colorama in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ipython>=4.0->ip
ythonblocks) (0.4.6)
Requirement already satisfied: decorator in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ipython>=4.0->i
pythonblocks) (5.2.1)
Requirement already satisfied: ipython-pygments-lexers in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from i
python>=4.0->ipythonblocks) (1.1.1)
Requirement already satisfied: jedi>=0.16 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ipython>=4.0->
ipythonblocks) (0.19.2)
Requirement already satisfied: matplotlib-inline in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ipython
>=4.0->ipythonblocks) (0.1.7)
Requirement already satisfied: prompt_toolkit<3.1.0,>=3.0.41 in d:\ai_journey\computational_intelligence\ci\lib\site-packages
(from ipython>=4.0->ipythonblocks) (3.0.52)
Requirement already satisfied: pygments>=2.4.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ipython>=
4.0->ipythonblocks) (2.19.2)
Requirement already satisfied: stack_data in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ipython>=4.0->
ipythonblocks) (0.6.3)
Requirement already satisfied: traitlets>=5.13.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ipython
>=4.0->ipythonblocks) (5.14.3)
Requirement already satisfied: wcwidth in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from prompt_toolkit<3.
1.0,>=3.0.41->ipython>=4.0->ipythonblocks) (0.2.13)
Requirement already satisfied: parso<0.9.0,>=0.8.4 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jedi>
=0.16->ipython>=4.0->ipythonblocks) (0.8.5)
Requirement already satisfied: jupyter-server<3,>=2.4.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from
notebook>=4.0->ipythonblocks) (2.17.0)
Requirement already satisfied: jupyterlab-server<3,>=2.27.1 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (f
rom notebook>=4.0->ipythonblocks) (2.27.3)
Requirement already satisfied: jupyterlab<4.5,>=4.4.5 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from no
tebook>=4.0->ipythonblocks) (4.4.7)
Requirement already satisfied: notebook-shim<0.3,>=0.2 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from n
otebook>=4.0->ipythonblocks) (0.2.4)
Requirement already satisfied: tornado>=6.2.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from notebook>=
4.0->ipythonblocks) (6.5.2)
Requirement already satisfied: anyio>=3.1.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyter-serv
er<3,>=2.4.0->notebook>=4.0->ipythonblocks) (4.10.0)
```

```
Requirement already satisfied: argon2-cffi>=21.1 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyter
-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (25.1.0)
Requirement already satisfied: jinja2>=3.0.3 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyter-ser
ver<3,>=2.4.0->notebook>=4.0->ipythonblocks) (3.1.6)
Requirement already satisfied: jupyter-client>=7.4.4 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jup
yter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (8.6.3)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (fro
m jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (5.8.1)
Requirement already satisfied: jupyter-events>=0.11.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ju
pyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (0.12.0)
Requirement already satisfied: jupyter-server-terminals>=0.4.4 in d:\ai_journey\computational_intelligence\ci\lib\site-packages
(from jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (0.5.3)
Requirement already satisfied: nbconvert>=6.4.4 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyter-
server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (7.16.6)
Requirement already satisfied: nbformat>=5.3.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyter-s
erver<3,>=2.4.0->notebook>=4.0->ipythonblocks) (5.10.4)
Requirement already satisfied: packaging>=22.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyter-s
erver<3,>=2.4.0->notebook>=4.0->ipythonblocks) (25.0)
Requirement already satisfied: prometheus-client>=0.9 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ju
pyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (0.22.1)
Requirement already satisfied: pywinpty>=2.0.1 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyter-s
erver<3,>=2.4.0->notebook>=4.0->ipythonblocks) (3.0.0)
Requirement already satisfied: pyzmq>=24 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyter-server<
3,>=2.4.0->notebook>=4.0->ipythonblocks) (27.0.2)
Requirement already satisfied: send2trash>=1.8.2 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyter
-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (1.8.3)
Requirement already satisfied: terminado>=0.8.3 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyter-
server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (0.18.1)
Requirement already satisfied: websocket-client>=1.7 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jup
yter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (1.8.0)
Requirement already satisfied: async-lru>=1.0.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyterl
ab<4.5,>=4.4.5->notebook>=4.0->ipythonblocks) (2.0.5)
Requirement already satisfied: httpx<1,>=0.25.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyterl
ab<4.5,>=4.4.5->notebook>=4.0->ipythonblocks) (0.28.1)
Requirement already satisfied: ipykernel!=6.30.0,>=6.5.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from
jupyterlab<4.5,>=4.4.5->notebook>=4.0->ipythonblocks) (6.30.1)
Requirement already satisfied: jupyter-lsp>=2.0.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyte
rlab<4.5,>=4.4.5->notebook>=4.0->ipythonblocks) (2.3.0)
Requirement already satisfied: setuptools>=41.1.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyte
rlab<4.5,>=4.4.5->notebook>=4.0->ipythonblocks) (80.9.0)
Requirement already satisfied: certifi in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from httpx<1,>=0.25.0-
```

```
>jupyterlab<4.5,>=4.4.5->notebook>=4.0->ipythonblocks) (2025.8.3)
Requirement already satisfied: httpcore==1.* in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from httpx<1,>=
0.25.0->jupyterlab<4.5,>=4.4.5->notebook>=4.0->ipythonblocks) (1.0.9)
Requirement already satisfied: idna in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from httpx<1,>=0.25.0->ju
pyterlab<4.5,>=4.4.5->notebook>=4.0->ipythonblocks) (3.10)
Requirement already satisfied: h11>=0.16 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from httpcore==1.*->
httpx<1,>=0.25.0->jupyterlab<4.5,>=4.4.5->notebook>=4.0->ipythonblocks) (0.16.0)
Requirement already satisfied: babel>=2.10 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyterlab-se
rver<3,>=2.27.1->notebook>=4.0->ipythonblocks) (2.17.0)
Requirement already satisfied: json5>=0.9.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyterlab-s
erver<3,>=2.27.1->notebook>=4.0->ipythonblocks) (0.12.1)
Requirement already satisfied: jsonschema>=4.18.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyte
rlab-server<3,>=2.27.1->notebook>=4.0->ipythonblocks) (4.25.1)
Requirement already satisfied: sniffio>=1.1 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from anyio>=3.1.0
->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (1.3.1)
Requirement already satisfied: typing_extensions>=4.5 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from an
yio>=3.1.0->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (4.15.0)
Requirement already satisfied: argon2-cffi-bindings in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from argo
n2-cffi>=21.1->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (25.1.0)
Requirement already satisfied: comm>=0.1.1 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ipykernel!=6.
30.0,>=6.5.0->jupyterlab<4.5,>=4.4.5->notebook>=4.0->ipythonblocks) (0.2.3)
Requirement already satisfied: debugpy>=1.6.5 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ipykernel!
=6.30.0,>=6.5.0->jupyterlab<4.5,>=4.4.5->notebook>=4.0->ipythonblocks) (1.8.16)
Requirement already satisfied: nest-asyncio>=1.4 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ipykern
el!=6.30.0,>=6.5.0->jupyterlab<4.5,>=4.4.5->notebook>=4.0->ipythonblocks) (1.6.0)
Requirement already satisfied: psutil>=5.7 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ipykernel!=6.
30.0,>=6.5.0->jupyterlab<4.5,>=4.4.5->notebook>=4.0->ipythonblocks) (7.0.0)
Requirement already satisfied: MarkupSafe>=2.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jinja2>=
3.0.3->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (3.0.2)
Requirement already satisfied: attrs>=22.2.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jsonschema>
=4.18.0->jupyterlab-server<3,>=2.27.1->notebook>=4.0->ipythonblocks) (25.3.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in d:\ai_journey\computational_intelligence\ci\lib\site-pac
kages (from jsonschema>=4.18.0->jupyterlab-server<3,>=2.27.1->notebook>=4.0->ipythonblocks) (2025.4.1)
Requirement already satisfied: referencing>=0.28.4 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jsons
chema>=4.18.0->jupyterlab-server<3,>=2.27.1->notebook>=4.0->ipythonblocks) (0.36.2)
Requirement already satisfied: rpds-py>=0.7.1 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jsonschema
>=4.18.0->jupyterlab-server<3,>=2.27.1->notebook>=4.0->ipythonblocks) (0.27.1)
Requirement already satisfied: python-dateutil>=2.8.2 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from ju
pyter-client>=7.4.4->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (2.9.0.post0)
Requirement already satisfied: platformdirs>=2.5 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyter
-core!=5.0.*,>=4.12->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (4.4.0)
```

```
Requirement already satisfied: pywin32>=300 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyter-cor
e!=5.0.*,>=4.12->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (311)
Requirement already satisfied: python-json-logger>=2.0.4 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from
jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (3.3.0)
Requirement already satisfied: pyyaml>=5.3 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyter-event
s>=0.11.0->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (6.0.2)
Requirement already satisfied: rfc3339-validator in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jupyter
-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (0.1.4)
Requirement already satisfied: rfc3986-validator>=0.1.1 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from
jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (0.1.1)
Requirement already satisfied: fqdn in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jsonschema[format-no
ngpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (1.5.1)
Requirement already satisfied: isoduration in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jsonschema[fo
rmat-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (20.11.0)
Requirement already satisfied: jsonpointer>1.13 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jsonsche
ma[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (3.0.0)
Requirement already satisfied: rfc3987-syntax>=1.1.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jso
nschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (1.1.0)
Requirement already satisfied: uri-template in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jsonschema[f
ormat-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (1.3.0)
Requirement already satisfied: webcolors>=24.6.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from jsonsch
ema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (24.11.1)
Requirement already satisfied: beautifulsoup4 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from nbconvert>
=6.4.4->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (4.13.5)
Requirement already satisfied: bleach!=5.0.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from bleach[cs
s]!=5.0.0->nbconvert>=6.4.4->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (6.2.0)
Requirement already satisfied: defusedxml in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from nbconvert>=6.
4.4->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (0.7.1)
Requirement already satisfied: jupyterlab-pygments in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from nbcon
vert>=6.4.4->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (0.3.0)
Requirement already satisfied: mistune<4,>=2.0.3 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from nbconve
rt>=6.4.4->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (3.1.4)
Requirement already satisfied: nbclient>=0.5.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from nbconvert
>=6.4.4->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (0.10.2)
Requirement already satisfied: pandocfilters>=1.4.1 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from nbco
nvert>=6.4.4->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (1.5.1)
Requirement already satisfied: webencodings in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from bleach!=5.0.
0->bleach[css]!=5.0.0->nbconvert>=6.4.4->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (0.5.1)
Requirement already satisfied: tinycss2<1.5,>=1.1.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from blea
ch[css]!=5.0.0->nbconvert>=6.4.4->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (1.4.0)
Requirement already satisfied: fastjsonschema>=2.15 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from nbfo
```

```
rmat>=5.3.0->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (2.21.2)
Requirement already satisfied: six>=1.5 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from python-dateutil>
=2.8.2->jupyter-client>=7.4.4->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (1.17.0)
Requirement already satisfied: charset_normalizer<4,>=2 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from
requests>=1.0->ipythonblocks) (3.4.3)
Requirement already satisfied: urllib3<3,>=1.21.1 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from reques
ts>=1.0->ipythonblocks) (2.5.0)
Requirement already satisfied: lark>=1.2.2 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from rfc3987-synta
x>=1.1.0->jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks)
(1.2.2)
Requirement already satisfied: cffi>=1.0.1 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from argon2-cffi-b
indings->argon2-cffi>=21.1->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (1.17.1)
Requirement already satisfied: pycparser in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from cffi>=1.0.1->ar
gon2-cffi-bindings->argon2-cffi>=21.1->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (2.22)
Requirement already satisfied: soupsieve>1.2 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from beautifulso
up4->nbconvert>=6.4.4->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (2.8)
Requirement already satisfied: arrow>=0.15.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from isoduration
->jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook>=4.0->ipythonblocks) (1.3.0)
Requirement already satisfied: types-python-dateutil>=2.8.10 in d:\ai_journey\computational_intelligence\ci\lib\site-packages
(from arrow>=0.15.0->isoduration->jsonschema[format-nongpl]>=4.18.0->jupyter-events>=0.11.0->jupyter-server<3,>=2.4.0->notebook
>=4.0->ipythonblocks) (2.9.0.20250822)
Requirement already satisfied: executing>=1.2.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from stack_da
ta->ipython>=4.0->ipythonblocks) (2.2.1)
Requirement already satisfied: asttokens>=2.1.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from stack_da
ta->ipython>=4.0->ipythonblocks) (3.0.0)
Requirement already satisfied: pure-eval in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from stack_data->ipy
thon>=4.0->ipythonblocks) (0.2.3)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: qpsolvers in d:\ai_journey\computational_intelligence\ci\lib\site-packages (4.8.1)
Requirement already satisfied: numpy>=1.15.4 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from qpsolvers)
(2.3.2)
Requirement already satisfied: scipy>=1.2.0 in d:\ai_journey\computational_intelligence\ci\lib\site-packages (from qpsolvers)
(1.16.1)
Note: you may need to restart the kernel to use updated packages.
```

Logical Sentences

Membuat Simbol dengan Kelas `Expr`

Tujuan: Memahami bagaimana cara merepresentasikan kalimat logika sederhana menggunakan kelas `Expr`. Tahap awal dimulai dengan mendefinisikan simbol, yang merupakan bentuk paling dasar dari ekspresi logika.

`Symbol('x')`
Penjelasan:

- `Symbol('x')` → perintah ini membuat sebuah simbol bernama `x`.
- Simbol adalah bentuk paling sederhana dari kelas `Expr`, dan digunakan sebagai representasi suatu variabel atau proposisi dalam logika.
- Setelah dijalankan, hasil keluaran yang ditampilkan adalah **x**, artinya simbol berhasil dibuat dan dapat digunakan untuk membentuk kalimat logika yang lebih kompleks.

In [214…
```
Symbol('x')
```

Out[214…
```
x
```

Mendefinisikan Beberapa Simbol Sekaligus

Tujuan: Mempelajari cara mendefinisikan lebih dari satu simbol logika dalam satu baris kode untuk mempersingkat penulisan.

`(x, y, P, Q, f) = symbols('x, y, P, Q, f')`
Penjelasan:

- `symbols('x, y, P, Q, f')` → perintah ini membuat beberapa simbol sekaligus, yaitu **x**, **y**, **P**, **Q**, dan **f**.
- Simbol **x** dan **y** biasanya dipakai sebagai variabel, sementara **P** dan **Q** digunakan sebagai proposisi logika.
- Simbol **f** dapat dipakai sebagai fungsi atau ekspresi lain sesuai kebutuhan.
- Dengan cara ini, kita tidak perlu menuliskan `Symbol('x')`, `Symbol('y')`, dan seterusnya satu per satu.

In [215…
```
(x, y, P, Q, f) = symbols('x, y, P, Q, f')
```

Menggabungkan Simbol dengan Operator Logika

Tujuan: Mencoba membentuk kalimat logika sederhana dengan menggunakan operator infix (`&`, `|`) dan prefix (`~`) yang tersedia di Python.

`P & ~Q`

Penjelasan:

- `P & ~Q` → membentuk kalimat logika yang artinya **P and not Q**.
- Tanda `&` digunakan untuk operasi **AND** (konjungsi).
- Tanda `~` digunakan untuk operasi **NOT** (negasi).
- Dengan demikian, agar ekspresi ini bernilai benar, maka proposisi **P** harus benar **dan** proposisi **Q** tidak benar.
- Hasil keluaran yang ditampilkan adalah bentuk logis sesuai definisi operator di kelas `Expr`.

`P | Q`

- `P | Q` → membentuk kalimat logika yang artinya **P or Q**.
- Tanda `|` digunakan untuk operasi **OR** (disjungsi).
- Ekspresi ini akan bernilai benar jika minimal salah satu dari proposisi **P** atau **Q** bernilai benar. Jika keduanya salah, maka hasilnya salah.

`~Q`

- `~Q` → membentuk kalimat logika yang artinya **not Q**.
- Tanda `~` digunakan untuk operasi negasi.
- Ekspresi ini akan bernilai benar hanya jika **Q bernilai salah**. Jika **Q benar**, maka hasilnya salah.

```
In [216…  P & ~Q
Out[216…  (P & ~Q)
In [217…  P | Q
Out[217…  (P | Q)
In [218…  ~Q
Out[218…  ~Q
```

Melihat Properti dari Objek `Expr`

Tujuan: Mengecek bagaimana sebuah kalimat logika disimpan di dalam objek `Expr`. Setiap objek `Expr` memiliki field seperti `op` (operator) dan `args` (argumen).

Penjelasan:

- `sentence = P & ~Q` → membuat sebuah ekspresi logika yang menyatakan **P and not Q**, lalu disimpan ke variabel bernama `sentence`.
- `sentence.op` → digunakan untuk melihat operator yang dipakai di dalam ekspresi tersebut.
- Karena ekspresinya adalah `P & ~Q`, maka nilai `sentence.op` yang ditampilkan adalah tanda **'&'**, yang berarti operator **AND**.
- `sentence.args` → menampilkan daftar argumen dari ekspresi `P & ~Q`. Argumen ini berupa tuple yang berisi dua elemen, yaitu `P` dan `~Q`. Jadi kita bisa tahu operatornya `&` dan argumennya adalah proposisi di kiri dan kanan.
- `P.args` → karena `P` adalah simbol tunggal, maka tidak memiliki argumen tambahan. Hasilnya adalah tuple kosong `()`.
- `Pxy = P(x, y)` → mendefinisikan ekspresi baru dengan menggunakan simbol `P` sebagai fungsi yang diberikan argumen `x` dan `y`. Dengan ini, `P` diperlakukan seperti predikat dalam logika predikat (contoh: `P(x,y)`).
- `Pxy.op` → menampilkan operator dari ekspresi `P(x, y)`, yaitu string `"P"`, karena `P` adalah nama predikat.
- `Pxy.args` → menampilkan tuple `(x, y)`, yaitu daftar argumen yang diberikan ke predikat `P`.

In [219…
```
sentence = P & ~Q

sentence.op
```

Out[219…
```
'&'
```

In [220…
```
sentence.args
```

Out[220…
```
(P, ~Q)
```

In [221…
```
P.op
```

Out[221…
```
'P'
```

In [222…
```
P.args
```

Out[222…
```
()
```

In [223…
```
Pxy = P(x, y)

Pxy.op
```

Out[223…
```
'P'
```

In [224…
```
Pxy.args
```

Out[224…
```
(x, y)
```

Mencoba Nested Expression dengan `Expr`

Tujuan: Melihat bagaimana kelas `Expr` dapat merepresentasikan ekspresi yang lebih kompleks, termasuk kombinasi angka, simbol, dan fungsi dalam bentuk pohon sintaks abstrak (abstract syntax tree).

```
3 * f(x, y) + P(y) / 2 + 1
```

Penjelasan:

- `3 * f(x, y)` → bagian ini mengalikan bilangan 3 dengan hasil dari fungsi `f(x, y)`. Di dalam `Expr`, hal ini akan direpresentasikan sebagai operator `*` dengan argumen `3` dan ekspresi `f(x, y)`.
- `P(y)` → ini adalah predikat `P` yang menerima argumen `y`. Sama seperti `P(x, y)` sebelumnya, hanya saja di sini predikatnya hanya punya satu argumen.
- `P(y) / 2` → hasil dari predikat `P(y)` dibagi dengan angka 2. Di dalam struktur `Expr`, operator yang dipakai adalah `/` dengan argumen `P(y)` dan `2`.
- `+ 1` → setelah semua operasi sebelumnya, ditambahkan angka `1`.
- Secara keseluruhan, ekspresi ini merupakan contoh **nested expression**, yaitu ekspresi yang berisi ekspresi lain di dalamnya. Hal ini menunjukkan bahwa `Expr` dapat digunakan untuk merepresentasikan struktur logika atau matematika yang lebih dalam dan kompleks.

In [225…
```
3 * f(x, y) + P(y) / 2 + 1
```

Out[225…
```
(((3 * f(x, y)) + (P(y) / 2)) + 1)
```

Operator untuk Membentuk Kalimat Logika

Berikut adalah penjelasan mengenai operator yang dapat digunakan untuk menyusun kalimat logika. Ada sedikit kendala: kita ingin menggunakan operator Python agar penulisan kalimat lebih sederhana, tetapi Python tidak mengizinkan penggunaan tanda panah implikasi secara langsung. Oleh karena itu, kita menggunakan notasi yang lebih panjang seperti `|'==>'|` sebagai pengganti `==>`. Alternatif lain, kita bisa menggunakan konstruktor `Expr` yang lebih eksplisit.

Daftar operator:

- **Negasi (¬P)**
  - Python Infix: `~P`
  - Hasil di Python: `~P`
  - Bentuk `Expr` : `Expr('~', P)`
- **Konjungsi (P ∧ Q)**
  - Python Infix: `P & Q`
  - Hasil di Python: `P & Q`
  - Bentuk `Expr` : `Expr('&', P, Q)`
- **Disjungsi (P ∨ Q)**
  - Python Infix: `P | Q`
  - Hasil di Python: `P | Q`
  - Bentuk `Expr` : `Expr('|', P, Q)`
- **Eksklusif Or / Ketidaksamaan (P ≠ Q)**
  - Python Infix: `P ^ Q`
  - Hasil di Python: `P ^ Q`
  - Bentuk `Expr` : `Expr('^', P, Q)`
- **Implikasi (P → Q)**
  - Python Infix: `P |'==>'| Q`
  - Hasil di Python: `P ==> Q`
  - Bentuk `Expr` : `Expr('==>', P, Q)`
- **Implikasi Terbalik (Q ← P)**

- Python Infix: `Q |'<==>'| P`
- Hasil di Python: `Q <==> P`
- Bentuk `Expr` : `Expr('<==>', Q, P)`

- **Ekuivalensi (P ↔ Q)**
  - Python Infix: `P |'<=>'| Q`
  - Hasil di Python: `P <=> Q`
  - Bentuk `Expr` : `Expr('<=>', P, Q)`

Contoh: Kita dapat mendefinisikan sebuah kalimat dengan menggunakan salah satu notasi di atas.

In [226…
```
~(P & Q)  |'==>'|  (~P | ~Q)
```

Out[226…
```
(~(P & Q) ==> (~P | ~Q))
```

In [227…
```
B |'<=='| A & ~B
```

Out[227…
```
(B <== (A & ~B))
```

In [228…
```
C ^ D
```

Out[228…
```
(C ^ D)
```

`expr` : Jalan Pintas untuk Membentuk Kalimat Logika

Tujuan: Mempermudah penulisan kalimat logika yang kompleks tanpa harus menggunakan notasi panjang seperti `|'==>'|`.

```
expr('~(P & Q)  ==>  (~P | ~Q)')
```

Penjelasan:

- Fungsi `expr` memungkinkan kita menuliskan kalimat logika dalam bentuk string yang lebih alami dan ringkas.
- `'~(P & Q)  ==>  (~P | ~Q)'` → menyatakan implikasi bahwa **jika tidak (P dan Q)**, maka **(~P atau ~Q)**.

- Hasil dari fungsi `expr` adalah sebuah objek `Expr` yang strukturnya sama seperti jika kita membentuk ekspresi tersebut dengan operator Python atau konstruktor `Expr`.
- Dengan cara ini, kode menjadi lebih mudah dibaca dan lebih dekat dengan notasi logika pada buku teks.

In [229…
```
expr('~(P & Q)  ==>  (~P | ~Q)')
```

Out[229…
```
(~(P & Q) ==> (~P | ~Q))
```

Menggunakan `expr` dengan Ekspresi Matematika

Tujuan: Menunjukkan bahwa fungsi `expr` tidak hanya dapat digunakan untuk kalimat logika, tetapi juga bisa dipakai untuk membentuk ekspresi matematika yang lebih umum.

`expr('sqrt(b ** 2 - 4 * a * c)')`
Penjelasan:

- `expr(...)` → menerima sebuah string, lalu mengubahnya menjadi objek `Expr`.
- `'sqrt(b ** 2 - 4 * a * c)'` → string ini merepresentasikan formula matematika klasik yaitu diskriminan pada persamaan kuadrat: $\sqrt{b^2 - 4ac}$.
- Operator `**` menunjukkan pangkat, sehingga `b ** 2` berarti b^2.
- Tanda `*` digunakan untuk perkalian, jadi `4 * a * c` berarti $4ac$.
- Hasil dari pemanggilan ini adalah sebuah objek `Expr` yang merepresentasikan ekspresi kuadrat tersebut dalam bentuk struktur data Python.

In [230…
```
expr('sqrt(b ** 2 - 4 * a * c)')
```

Out[230…
```
sqrt(((b ** 2) - ((4 * a) * c)))
```

Basis Pengetahuan Proposisional: `PropKB`

Kelas `PropKB` digunakan untuk merepresentasikan **basis pengetahuan** yang berisi kalimat-kalimat logika proposisional.

Dalam hierarki kelas, sebenarnya ada kelas induk bernama `KB` yang menyediakan empat metode utama selain konstruktor `__init__`. Salah satu hal penting adalah bahwa metode `ask` pada dasarnya hanya memanggil `ask_generator`. Jadi, jika kita ingin membuat kelas basis pengetahuan baru, biasanya yang perlu diimplementasikan hanyalah `ask_generator`, bukan `ask`.

Berikut adalah penjelasan fungsi-fungsi dalam `PropKB`:

- **`__init__(self, sentence=None)`** Konstruktor ini membuat satu atribut utama bernama `clauses`, yaitu sebuah list yang menyimpan semua kalimat dalam basis pengetahuan. Setiap kalimat diubah menjadi bentuk **clause** (kalimat yang terdiri dari literal dan operator `or`).
- **`tell(self, sentence)`** Fungsi ini digunakan untuk **menambahkan** kalimat baru ke dalam basis pengetahuan. Prosesnya otomatis: kalimat diubah ke bentuk **CNF (Conjunctive Normal Form)**, kemudian semua clause hasil konversi dimasukkan ke dalam atribut `clauses`. Dengan demikian, kita tidak perlu repot menulis kalimat langsung dalam bentuk clause.
- **`ask_generator(self, query)`** Metode ini adalah inti dari proses **penarikan kesimpulan**. Di dalamnya dipanggil fungsi `tt_entails`, yang akan mengembalikan `True` jika basis pengetahuan memenuhi query, dan `False` jika tidak.
  - Jika entailed → fungsi ini mengembalikan dictionary kosong `{}`.
  - Jika tidak entailed → fungsi ini mengembalikan `None`. Alasan penggunaan `{}` dan `None` (bukan sekadar `True` / `False`) adalah untuk konsistensi dengan logika tingkat pertama (First-Order Logic), di mana fungsi `ask_generator` seharusnya mengembalikan semua substitusi yang membuat query bernilai benar.
- **`retract(self, sentence)`** Fungsi ini digunakan untuk **menghapus** kalimat dari basis pengetahuan. Sama seperti `tell`, input bisa berupa kalimat dalam bentuk apapun. Fungsi ini akan otomatis mengonversi kalimat menjadi clause, lalu menghapus clause tersebut dari atribut `clauses`.

Membuat Knowledge Base untuk Wumpus World

Tujuan: Membangun sebuah **basis pengetahuan proposisional** (PropKB) khusus untuk dunia Wumpus. Basis pengetahuan ini nantinya akan diisi dengan kalimat-kalimat logika yang menggambarkan aturan serta kondisi pada lingkungan Wumpus World.

```
wumpus_kb = PropKB()
```

Penjelasan:

- `wumpus_kb = PropKB()` → membuat sebuah objek `PropKB` kosong yang dinamai `wumpus_kb`.

- Objek ini akan berfungsi sebagai **wadah** untuk menyimpan semua pengetahuan logika proposisional yang terkait dengan dunia Wumpus.
- Selanjutnya, kita dapat menambahkan kalimat logika (misalnya aturan tentang keberadaan Wumpus, bau, atau lubang) ke dalam knowledge base ini dengan menggunakan metode `tell`.

In [231…
```
wumpus_kb = PropKB()
```

Mendefinisikan Simbol untuk Aturan Wumpus World

Tujuan: Membuat simbol-simbol proposisional yang mewakili kondisi tertentu di dunia Wumpus. Simbol ini digunakan dalam clause untuk menggambarkan apakah ada lubang (pit) atau angin (breeze) pada koordinat tertentu.

```
P11, P12, P21, P22, P31, B11, B21 = expr('P11, P12, P21, P22, P31, B11, B21')
```
Penjelasan:

- $P_{x,y}$ → bernilai benar jika terdapat **pit (lubang)** pada posisi $[x, y]$.
- $B_{x,y}$ → bernilai benar jika agen merasakan adanya **breeze (angin)** pada posisi $[x, y]$.
- `expr('P11, P12, P21, P22, P31, B11, B21')` → membuat tujuh simbol proposisional sekaligus:
  - **P11**: ada pit di sel (1,1).
  - **P12**: ada pit di sel (1,2).
  - **P21**: ada pit di sel (2,1).
  - **P22**: ada pit di sel (2,2).
  - **P31**: ada pit di sel (3,1).
  - **B11**: ada breeze di sel (1,1).
  - **B21**: ada breeze di sel (2,1).

Simbol-simbol ini nantinya akan dipakai untuk menyusun aturan logika (clauses) dalam knowledge base `wumpus_kb`.

In [232…
```
P11, P12, P21, P22, P31, B11, B21 = expr('P11, P12, P21, P22, P31, B11, B21')
```

Menambahkan Fakta ke Knowledge Base

Tujuan: Memasukkan informasi baru ke dalam basis pengetahuan Wumpus World. Informasi ini berupa fakta yang diketahui dari kondisi awal lingkungan.

`wumpus_kb.tell(~P11)`
Penjelasan:

- `~P11` → menyatakan **tidak ada pit (lubang) di sel (1,1)**.
- `wumpus_kb.tell(~P11)` → menambahkan fakta tersebut ke dalam knowledge base `wumpus_kb`.
- Metode `tell` akan otomatis mengubah kalimat ke bentuk CNF (jika diperlukan) dan menyimpannya dalam daftar klausa.
- Dengan menambahkan informasi ini, agen akan tahu dengan pasti bahwa lokasi awal `(1,1)` aman dari lubang.

In [233…
```
wumpus_kb.tell(~P11)
```

Menambahkan Aturan Breeze ke Knowledge Base

Tujuan: Mendefinisikan hubungan antara **breeze** dan **pit** pada sel tetangga. Aturan ini menyatakan bahwa sebuah kotak akan terasa berangin (*breezy*) **jika dan hanya jika** ada pit di salah satu kotak yang berdekatan.

```
wumpus_kb.tell(B11 | '<=>' | ((P12 | P21)))
wumpus_kb.tell(B21 | '<=>' | ((P11 | P22 | P31)))
```
Penjelasan:

- `B11 | '<=>' | (P12 | P21)` → berarti **sel (1,1) berangin jika dan hanya jika ada pit di (1,2) atau (2,1)**.
- `B21 | '<=>' | (P11 | P22 | P31)` → berarti **sel (2,1) berangin jika dan hanya jika ada pit di (1,1), (2,2), atau (3,1)**.
- Operator `<=>` menyatakan **ekuivalensi logis (if and only if)**, sehingga kedua arah hubungan berlaku.
- Dengan menambahkan aturan ini, agen akan dapat menyimpulkan keberadaan pit berdasarkan ada atau tidaknya breeze yang dirasakan.

In [234…
```
wumpus_kb.tell(B11 | '<=>' | ((P12 | P21)))
wumpus_kb.tell(B21 | '<=>' | ((P11 | P22 | P31)))
```

Menambahkan Informasi Percept Breeze

Tujuan: Memasukkan data percept yang diterima agen pada beberapa kotak awal. Percept ini menunjukkan apakah agen merasakan breeze di lokasi tertentu. Informasi ini akan membantu agen dalam melakukan penalaran tentang posisi pit di sekitar.

```
wumpus_kb.tell(~B11)
wumpus_kb.tell(B21)
```

Penjelasan:

- `~B11` → menyatakan bahwa **tidak ada breeze di sel (1,1)**. Hal ini logis karena sel awal biasanya aman.
- `B21` → menyatakan bahwa **ada breeze di sel (2,1)**. Artinya, di salah satu kotak tetangga sel (2,1) kemungkinan terdapat pit.
- Dengan menambahkan kedua fakta ini ke knowledge base, agen memiliki data nyata (percept) untuk dipadukan dengan aturan logika sebelumnya.
- Informasi ini sangat penting untuk melakukan inference, misalnya menentukan kemungkinan ada pit di `(2,2)` atau `(3,1)`.

In [235…
```
wumpus_kb.tell(~B11)
wumpus_kb.tell(B21)
```

Mengecek Klausa dalam Knowledge Base

Tujuan: Melihat isi dari basis pengetahuan yang sudah dibangun sejauh ini. Hal ini dilakukan untuk memastikan bahwa semua fakta dan aturan yang ditambahkan benar-benar tersimpan di dalam knowledge base.

```
wumpus_kb.clauses
```

Penjelasan:

- `wumpus_kb.clauses` → digunakan untuk menampilkan semua klausa yang ada di dalam objek `wumpus_kb`.
- Setiap klausa yang ditampilkan adalah hasil konversi dari kalimat logika yang sebelumnya ditambahkan menggunakan `tell()`.
- Dengan cara ini, kita bisa memverifikasi bahwa aturan breeze, fakta tentang pit, dan percept sudah tercatat dalam bentuk klausa.
- Pengecekan ini penting sebelum melakukan query (`ask`) agar kita yakin basis pengetahuan berisi semua informasi yang diperlukan untuk inference.

In [236…
```
wumpus_kb.clauses
```

Out[236...
```
[~P11,
 (~P12 | B11),
 (~P21 | B11),
 (P12 | P21 | ~B11),
 (~P11 | B21),
 (~P22 | B21),
 (~P31 | B21),
 (P11 | P22 | P31 | ~B21),
 ~B11,
 B21]
```

Kita melihat bahwa ekuivalensi $B_{1,1} \iff (P_{1,2} \vee P_{2,1})$ secara otomatis dikonversi menjadi dua implikasi yang kemudian diubah ke bentuk CNF dan disimpan dalam `KB`.

- $B_{1,1} \iff (P_{1,2} \vee P_{2,1})$ dipecah menjadi $B_{1,1} \implies (P_{1,2} \vee P_{2,1})$ dan $B_{1,1} \impliedby (P_{1,2} \vee P_{2,1})$.
- $B_{1,1} \implies (P_{1,2} \vee P_{2,1})$ dikonversi menjadi $P_{1,2} \vee P_{2,1} \vee \neg B_{1,1}$.
- $B_{1,1} \impliedby (P_{1,2} \vee P_{2,1})$ dikonversi menjadi $\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1}$ yang kemudian berubah menjadi $(\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$ setelah menerapkan hukum De Morgan dan distribusi disjungsi.
- $B_{2,1} \iff (P_{1,1} \vee P_{2,2} \vee P_{3,2})$ dikonversi dengan cara yang serupa.

Knowledge-Based Agents

Tujuan: Memahami konsep agen berbasis pengetahuan, yaitu agen yang menggunakan **knowledge base (KB)** untuk menyimpan informasi, melakukan inferensi, dan menentukan tindakan terbaik.

Penjelasan:

- Agen berbasis pengetahuan adalah agen generik sederhana yang **menyimpan** serta **mengelola** basis pengetahuan.
- Pada awalnya, knowledge base bisa saja sudah berisi pengetahuan awal (background knowledge).
- Fungsi utama agen berbasis KB adalah menyediakan **abstraksi** dalam manipulasi knowledge base. Dengan abstraksi ini, agen dapat menerima percept, memperbarui pengetahuannya, melakukan query untuk menentukan tindakan terbaik, dan kemudian mencatat tindakan yang sudah dilakukan.
- Implementasi `KB-Agent` dalam kode ini terdapat pada kelas **`KBAgentProgram`**, yang diturunkan dari kelas `KB`.

- Kelas ini berfungsi untuk menghubungkan percept yang diterima agen dengan pengetahuan di dalam KB, lalu menghasilkan aksi yang sesuai.

`psource(KBAgentProgram)`
Kode di atas digunakan untuk menampilkan source code dari kelas `KBAgentProgram`, sehingga kita bisa mempelajari detail implementasinya.

In [237…
```
psource(KBAgentProgram)
```

```
def KBAgentProgram(kb):
    """
    [Figure 7.1]
    A generic logical knowledge-based agent program.
    """
    steps = itertools.count()

    def program(percept):
        t = next(steps)
        kb.tell(make_percept_sentence(percept, t))
        action = kb.ask(make_action_query(t))
        kb.tell(make_action_sentence(action, t))
        return action

    def make_percept_sentence(percept, t):
        return Expr('Percept')(percept, t)

    def make_action_query(t):
        return expr('ShouldDo(action, {})'.format(t))

    def make_action_sentence(action, t):
        return Expr('Did')(action[expr('action')], t)

    return program
```

Fungsi bantu `make_percept_sentence`, `make_action_query`, dan `make_action_sentence` masing-masing telah dinamai sesuai dengan fungsinya, yaitu:

- `make_percept_sentence` → membuat kalimat logika orde pertama mengenai percept yang ingin diterima oleh agen.
- `make_action_query` → menanyakan kepada **KB** tentang tindakan apa yang seharusnya diambil.

- `make_action_sentence` → memberi tahu **KB** mengenai tindakan yang baru saja dilakukan oleh agen.

Inferensi dalam Basis Pengetahuan Proposisional

Tujuan: Mempelajari cara menentukan apakah sebuah kalimat logika di-*entail* oleh sebuah knowledge base (KB). Dengan kata lain, kita ingin memutuskan apakah $KB \models \alpha$ untuk suatu kalimat α.

Truth Table Enumeration

Penjelasan:

- Metode ini menggunakan pendekatan **model checking**.
- Semua kemungkinan model dieksplorasi, yaitu semua kombinasi nilai benar/salah dari simbol-simbol proposisional di dalam KB.
- Untuk setiap model, dicek apakah KB bernilai benar. Jika benar, maka dicek juga apakah α bernilai benar di model tersebut.
- Jika dalam semua model di mana KB benar, α juga benar, maka dapat disimpulkan bahwa $KB \models \alpha$.
- Jumlah model yang harus dicek adalah 2^n, dengan n = jumlah simbol dalam KB.

`psource(tt_check_all)`
Kode di atas digunakan untuk menampilkan implementasi fungsi `tt_check_all`, yaitu fungsi yang melakukan pengecekan truth table enumeration.

In [238…
```
psource(tt_check_all)
```

```
def tt_check_all(kb, alpha, symbols, model):
    """Auxiliary routine to implement tt_entails."""
    if not symbols:
        if pl_true(kb, model):
            result = pl_true(alpha, model)
            assert result in (True, False)
            return result
        else:
            return True
    else:
        P, rest = symbols[0], symbols[1:]
        return (tt_check_all(kb, alpha, rest, extend(model, P, True)) and
                tt_check_all(kb, alpha, rest, extend(model, P, False)))
```

Algoritma ini pada dasarnya menghitung setiap baris dari tabel kebenaran $KB \implies \alpha$ dan memeriksa apakah ekspresi tersebut benar di semua baris.

Jika simbol-simbol sudah didefinisikan, prosedur akan secara rekursif membangun semua kombinasi nilai kebenaran untuk simbol-simbol tersebut, lalu memeriksa apakah `model` konsisten dengan `kb`. Model-model yang diperoleh sesuai dengan baris-baris pada tabel kebenaran yang memiliki nilai `true` pada kolom KB, dan pada baris tersebut kemudian diperiksa apakah query juga bernilai benar.

`result = pl_true(alpha, model)`.

Secara singkat, `tt_check_all` mengevaluasi ekspresi logika berikut untuk setiap `model`:

`pl_true(kb, model) => pl_true(alpha, model)`

yang secara logis ekuivalen dengan:

`pl_true(kb, model) & ~pl_true(alpha, model)`

Artinya, knowledge base dan negasi dari query bersifat tidak konsisten secara logis.

Fungsi `tt_entails()` hanya mengambil simbol-simbol dari query, lalu memanggil `tt_check_all()` dengan parameter yang sesuai.

In [239…

```
psource(tt_entails)
```

```
def tt_entails(kb, alpha):
    """
    [Figure 7.10]
    Does kb entail the sentence alpha? Use truth tables. For propositional
    kb's and sentences. Note that the 'kb' should be an Expr which is a
    conjunction of clauses.
    >>> tt_entails(expr('P & Q'), expr('Q'))
    True
    """
    assert not variables(alpha)
    symbols = list(prop_symbols(kb & alpha))
    return tt_check_all(kb, alpha, symbols, {})
```

Contoh Penggunaan `tt_entails()`

Tujuan: Menunjukkan bagaimana fungsi `tt_entails` bekerja dalam kasus sederhana dengan simbol proposisional P dan Q.

`tt_entails(P & Q, Q)`
Penjelasan:

- `tt_entails(P & Q, Q)` → memeriksa apakah dari knowledge base $P \wedge Q$, dapat disimpulkan bahwa Q bernilai benar.
- Secara logika, jika $P \wedge Q$ benar, maka otomatis Q juga benar.
- Output yang diberikan adalah **True**, artinya $KB \models Q$.
- Hal ini sesuai dengan aturan dasar logika proposisional: sebuah konjungsi ($P \wedge Q$) akan meng-*entail* masing-masing komponennya (P dan Q).

In [240…
```
tt_entails(P & Q, Q)
```

Out[240…
```
True
```

Contoh `tt_entails` dengan Disjungsi

Tujuan: Mengilustrasikan bagaimana `tt_entails` bekerja saat basis pengetahuan berupa disjungsi ($P \vee Q$).

`tt_entails(P | Q, Q)`
Penjelasan:

- `tt_entails(P | Q, Q)` → memeriksa apakah dari knowledge base $P \vee Q$, dapat disimpulkan bahwa Q benar.
- Hasil yang diberikan adalah **False**.
- Alasannya: dari $P \vee Q$, kita hanya tahu **salah satu** dari P atau Q benar, tetapi tidak dapat memastikan bahwa Q selalu benar.
- Contoh: jika $P = True$ dan $Q = False$, maka $P \vee Q = True$, tetapi query Q bernilai False.
- Maka, $KB = (P \vee Q)$ **tidak meng-entail** Q.
- `tt_entails(P | Q, P)` → memeriksa apakah dari knowledge base $P \vee Q$, dapat disimpulkan bahwa P bernilai benar.
- Hasilnya adalah **False**.
- Alasannya mirip dengan kasus sebelumnya: dari $P \vee Q$, kita tahu paling tidak salah satu dari P atau Q benar, tetapi tidak bisa dipastikan bahwa P selalu benar.
- Contoh: jika $P = False$ dan $Q = True$, maka $P \vee Q = True$, tetapi query P bernilai False.
- Jadi, $KB = (P \vee Q)$ tidak meng-entail P.

In [241…
```
tt_entails(P | Q, Q)
```

Out[241…
```
False
```

In [242…
```
tt_entails(P | Q, P)
```

Out[242…
```
False
```

Contoh Kompleks dengan `tt_entails`

Tujuan: Menguji entailment pada kasus yang lebih rumit dengan banyak simbol proposisional.

```
(A, B, C, D, E, F, G) = symbols('A, B, C, D, E, F, G')
tt_entails(A & (B | C) & D & E & ~(F | G), A & D & E & ~F & ~G)
```

Penjelasan:

- Pertama, dibuat tujuh simbol proposisional: **A, B, C, D, E, F, G**.
- Basis pengetahuan (KB) yang digunakan:

  $$A \land (B \lor C) \land D \land E \land \neg(F \lor G)$$

  Artinya:

  - A, D, dan E bernilai benar.
  - Minimal salah satu dari B atau C bernilai benar.
  - Baik F maupun G bernilai salah.
- Query yang dicek:

  $$A \land D \land E \land \neg F \land \neg G$$

- Karena semua kondisi di query sudah tercakup dalam KB (ditambah KB punya informasi ekstra tentang $B \lor C$), hasil entailment adalah **True**.
- Dengan kata lain, query tersebut memang konsisten dan dapat disimpulkan dari knowledge base yang ada.

In [243…
```
(A, B, C, D, E, F, G) = symbols('A, B, C, D, E, F, G')
tt_entails(A & (B | C) & D & E & ~(F | G), A & D & E & ~F & ~G)
```

Out[243…
```
True
```

Kita dapat melihat bahwa agar **KB** bernilai benar, A, D, dan E harus bernilai **True**, sedangkan F dan G harus bernilai **False**. Tidak ada kesimpulan yang dapat diambil mengenai nilai kebenaran B atau C.

Kembali ke permasalahan kita, perlu dicatat bahwa `tt_entails()` menerima sebuah `Expr` yang berupa konjungsi dari klausa sebagai input, bukan langsung objek `KB` itu sendiri. Kita dapat menggunakan metode `ask_if_true()` dari `PropKB` yang akan melakukan semua konversi yang diperlukan secara otomatis. Sekarang mari kita periksa apa yang dapat diberitahukan oleh `wumpus_kb` mengenai $P_{1,1}$.

In [244…
```
wumpus_kb.ask_if_true(~P11), wumpus_kb.ask_if_true(P11)
```

Out[244…
```
(True, False)
```

Menggunakan `ask_if_true()` pada Simbol Lain

Tujuan: Menguji apakah basis pengetahuan Wumpus dapat menyimpulkan kebenaran atau kesalahan untuk proposisi tertentu selain $P_{1,1}$, misalnya $P_{2,2}$ dan $P_{3,1}$.

`wumpus_kb.ask_if_true(~P22), wumpus_kb.ask_if_true(P22)`
Penjelasan:

- Hasil yang diperoleh adalah **(False, False)**.
- Artinya, dari knowledge base yang ada, sistem **tidak bisa memastikan** apakah $P_{2,2}$ bernilai benar atau salah.
- `ask_if_true(~P22)` mengembalikan **False** → tidak ada cukup informasi untuk menyimpulkan bahwa di sel (2,2) pasti **tidak ada pit**.
- `ask_if_true(P22)` juga mengembalikan **False** → tidak ada cukup informasi untuk menyimpulkan bahwa di sel (2,2) pasti **ada pit**.
- Jadi, pada tahap ini agen hanya tahu bahwa $P_{2,2}$ **masih tidak pasti** (unknown) berdasarkan fakta dan aturan yang sudah dimasukkan.

In [245…
```
wumpus_kb.ask_if_true(~P22), wumpus_kb.ask_if_true(P22)
```

Out[245…
```
(False, False)
```

KJkladsjfkajfklajdfaklsfjdsklajfadklsdfjaskljfas