**Philosophical Cognitive Architecture - Installation Guide**

**Overview**

A working implementation of a cognitive architecture with:

* ✅ De se (self-locating) belief system
* ✅ Formal safety verification using Z3
* ✅ Propositional attitudes (Chalmers framework)
* ✅ Runtime safety monitoring
* ✅ Complete working examples

**Installation**

**1. Prerequisites**

# Python 3.10 or higher

python --version

# Create virtual environment

python -m venv venv

source venv/bin/activate # On Windows: venv\Scripts\activate

**2. Install Dependencies**

pip install z3-solver>=4.12.0

pip install numpy>=1.24.0

**3. Project Structure**

philosophical\_agi/

├── core/

│ ├── \_\_init\_\_.py

│ ├── types.py # Data structures

│ ├── belief\_system.py # Belief management

│ ├── agent.py # Main agent

│ └── memory.py # Memory system (optional)

├── safety/

│ ├── \_\_init\_\_.py

│ ├── verifier.py # Z3 formal verification

│ └── monitor.py # Runtime monitoring

└── examples/

└── demo.py # Working demonstrations

**4. Create Project Files**

Copy the code from the main artifact into the appropriate files according to the structure above.

For \_\_init\_\_.py files:

**philosophical\_agi/core/init.py:**

from .types import (

CenteredWorld,

PropositionalAttitude,

AttitudeType,

WorkspaceBounds,

AgentState,

Action

)

from .belief\_system import DeSeBeliefSystem

from .agent import CognitiveAgent

\_\_all\_\_ = [

'CenteredWorld',

'PropositionalAttitude',

'AttitudeType',

'WorkspaceBounds',

'AgentState',

'Action',

'DeSeBeliefSystem',

'CognitiveAgent'

]

**philosophical\_agi/safety/init.py:**

from .verifier import FormalSafetyVerifier

from .monitor import RuntimeSafetyMonitor

\_\_all\_\_ = [

'FormalSafetyVerifier',

'RuntimeSafetyMonitor'

]

**Running the Demonstrations**

**Basic Demo**

cd philosophical\_agi

python examples/demo.py

Expected output:

======================================================================

COGNITIVE ARCHITECTURE DEMONSTRATION

======================================================================

✅ CognitiveAgent 'DemoAgent' initialized

Workspace: [0.0, 10.0] × [0.0, 10.0]

--- Running 5 cognitive cycles ---

Cycle 1:

Position: (5.0, 5.0)

Action: EXPLORE

Target: (5.2, 4.8)

Beliefs: 3

✓ Action verified safe

...

**Testing**

**Manual Tests**

Create tests/test\_agent.py:

import sys

sys.path.insert(0, '..')

from philosophical\_agi.core import CognitiveAgent, WorkspaceBounds

from philosophical\_agi.core.types import Action

def test\_agent\_creation():

"""Test basic agent creation"""

agent = CognitiveAgent("TestAgent")

assert agent.id == "TestAgent"

assert agent.time == 0

print("✓ Agent creation test passed")

def test\_belief\_system():

"""Test belief updates"""

agent = CognitiveAgent("TestAgent")

# Add de se belief

agent.beliefs.add\_de\_se\_belief('location', (1.0, 2.0, 0.0))

beliefs = agent.beliefs.get\_all\_beliefs()

assert len(beliefs) > 0

assert any('SELF\_location' in b for b in beliefs)

print("✓ Belief system test passed")

def test\_safety\_verification():

"""Test safety verification"""

agent = CognitiveAgent("TestAgent")

# Test safe action

safe\_action = Action(

action\_type="MOVE",

target\_position=(5.0, 5.0, 0.0)

)

result, info = agent.safety\_monitor.monitor\_action(agent.state, safe\_action)

assert not info['modified']

print("✓ Safe action test passed")

# Test unsafe action (out of bounds)

bounds = WorkspaceBounds(x\_min=0, x\_max=10)

agent.state.workspace\_bounds = bounds

unsafe\_action = Action(

action\_type="MOVE",

target\_position=(15.0, 5.0, 0.0)

)

result, info = agent.safety\_monitor.monitor\_action(agent.state, unsafe\_action)

assert info['modified']

print("✓ Unsafe action detection test passed")

def test\_human\_proximity():

"""Test human proximity safety"""

agent = CognitiveAgent("TestAgent")

agent.state.human\_positions = [(5.0, 5.0, 0.0)]

# Too close to human

action = Action(

action\_type="MOVE",

target\_position=(5.5, 5.0, 0.0)

)

result, info = agent.safety\_monitor.monitor\_action(agent.state, action)

assert info['modified']

print("✓ Human proximity test passed")

if \_\_name\_\_ == "\_\_main\_\_":

test\_agent\_creation()

test\_belief\_system()

test\_safety\_verification()

test\_human\_proximity()

print("\n✅ All tests passed!")

Run tests:

cd tests

python test\_agent.py

**Key Features Explained**

**1. De Se (Self-Locating) Beliefs**

The system implements Perry's distinction between:

* **De dicto**: "Someone is in danger" (general fact)
* **De se**: "I am in danger" (self-locating, triggers action)

# De dicto belief

agent.beliefs.add\_de\_dicto\_belief("weather is rainy")

# De se belief (action-triggering)

agent.beliefs.add\_de\_se\_belief('in\_danger', True, confidence=0.9)

**2. Formal Safety Verification**

Uses Z3 theorem prover for mathematical proofs:

# The verifier proves:

# ∀ actions: safe(action) ↔

# within\_bounds(action) ∧

# ∀ humans: distance(action, human) ≥ 1.0m ∧

# ∀ obstacles: distance(action, obstacle) ≥ radius

**3. Propositional Attitudes**

Implements Chalmers' framework:

* **Beliefs**: Veridicality-assessable states
* **Desires**: Goal-directed states
* **Intentions**: Action commitments
* **Credences**: Probabilistic beliefs

# Add different attitude types

agent.beliefs.add\_desire("reach\_goal", urgency=0.8)

agent.beliefs.add\_de\_se\_belief('location', position, confidence=0.95)

**4. Runtime Safety Monitoring**

Two-layer safety:

1. **Fast heuristics**: Quick checks (NaN, bounds, velocity)
2. **Formal verification**: Z3 proofs for mathematical guarantees

**Advanced Usage**

**Custom Workspace**

bounds = WorkspaceBounds(

x\_min=-5.0, x\_max=5.0,

y\_min=-5.0, y\_max=5.0,

z\_min=0.0, z\_max=2.0

)

agent = CognitiveAgent("CustomAgent", workspace\_bounds=bounds)

**Adding Obstacles**

agent.state.obstacles = [

{'position': (3.0, 3.0, 0.0), 'radius': 1.0},

{'position': (7.0, 7.0, 0.0), 'radius': 0.5}

]

**Monitoring Safety Statistics**

thought\_log = agent.get\_thought\_log()

stats = thought\_log['safety\_stats']

print(f"Total violations: {stats['total\_violations']}")

print(f"Success rate: {stats['verifier\_stats']['success\_rate']:.1%}")

**Troubleshooting**

**Z3 Import Error**

pip install --upgrade z3-solver

**Performance Issues**

For faster execution, reduce formal verification frequency:

# Only verify critical actions

if action.action\_type in ['MOVE', 'GRASP']:

safe\_action, info = agent.safety\_monitor.monitor\_action(state, action)

**Next Steps**

1. **Add Learning**: Integrate reinforcement learning for action selection
2. **Extend Beliefs**: Add temporal reasoning and belief revision
3. **Hardware Integration**: Connect to robotics platforms (PyBullet, ROS)
4. **Multi-Agent**: Extend to multiple agents with social reasoning

**Documentation**

* Code is heavily commented
* Each class has docstrings
* Type hints throughout for clarity

**License**

MIT License - Feel free to use and modify

**Citation**

If you use this in research:

@software{philosophical\_agi\_2024,

title={Philosophical Cognitive Architecture with Formal Safety},

author={Your Name},

year={2024},

note={Working implementation of de se reasoning with Z3 verification}

}