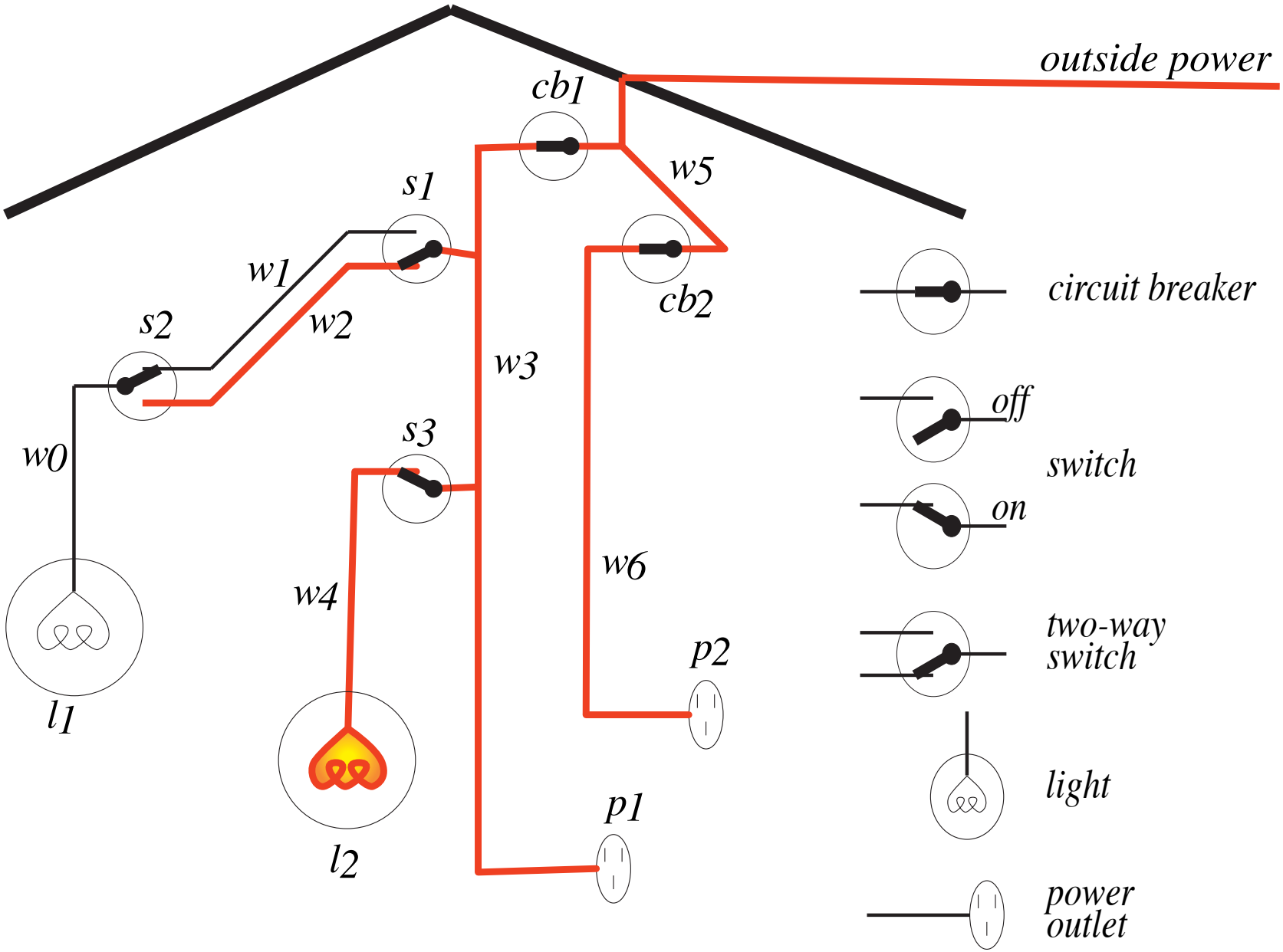
Electrical Domain



Users

- In the electrical domain, what should the house builder know?
- What should an occupant know?

Users

- In the electrical domain, what should the house builder know?
- What should an occupant know?
- Users can't be expected to volunteer knowledge:

Users

- In the electrical domain, what should the house builder know?
- What should an occupant know?
- Users can't be expected to volunteer knowledge:
  - They don't know what information is needed.
  - They don't know what vocabulary to use.

Ask-the-user

- Users can provide observations to the system.

Ask-the-user

- Users can provide observations to the system.
- They typically don't know what information is useful, and don't know the syntax or terminology to use.

Ask-the-user

- Users can provide observations to the system.
- They typically don't know what information is useful, and don't know the syntax or terminology to use.
- They can answer questions.

Ask-the-user

- Users can provide observations to the system.
- They typically don't know what information is useful, and don't know the syntax or terminology to use.
- They can answer questions.
- Askable atoms are those that a user should be able to provide a truth value for.

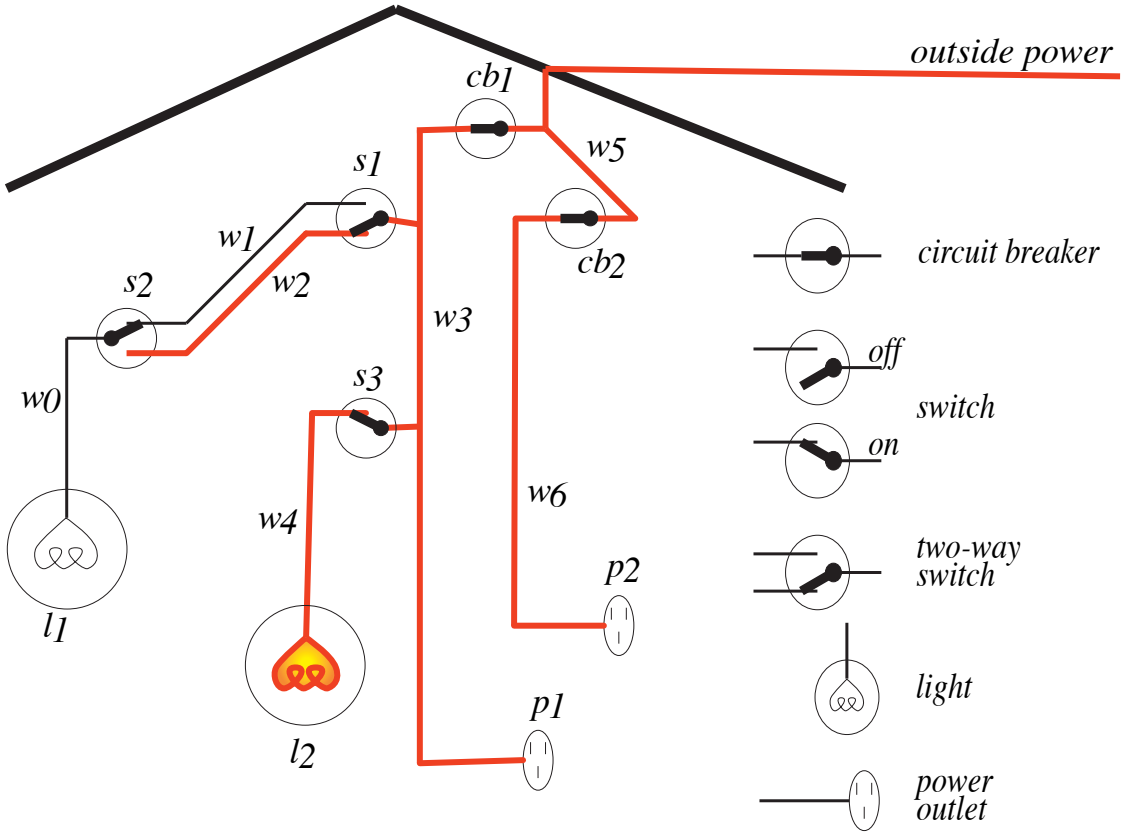
Ask-the-user

- Users can provide observations to the system.
- They typically don't know what information is useful, and don't know the syntax or terminology to use.
- They can answer questions.
- Askable atoms are those that a user should be able to provide a truth value for.
- There are 3 sorts of atoms to be proved in the top-down proof procedure:
  - how for which the user isn't expected to know the answer
  - askable atoms that may be useful in the proof
  - askable atoms that the user has already provided information about.

Ask-the-user

- Users can provide observations to the system.
- They typically don't know what information is useful, and don't know the syntax or terminology to use.
- They can answer questions.
- Askable atoms are those that a user should be able to provide a truth value for.
- There are 3 sorts of atoms to be proved in the top-down proof procedure:
  - how for which the user isn't expected to know the answer
  - askable atoms that may be useful in the proof
  - askable atoms that the user has already provided information about.
- The top-down proof procedure can be modified to ask users about askable atoms they have not already provided answers for.

Electrical Environment (aipython.org interaction)




```
python -i logicExplain.py
interact(elect)
ask lit_l1
```

Knowledge-Level Explanation

- HOW questions can be used to ask how an atom was proved.
  It gives the rule used to prove the atom.
  A user can ask HOW an element of the body of that rules was proved.
  This lets the user explore a proof.

Knowledge-Level Explanation

- HOW questions can be used to ask how an atom was proved.
  It gives the rule used to prove the atom.
  A user can ask HOW an element of the body of that rules was proved.
  This lets the user explore a proof.
  (Continuing the aipython session):

  ```
  ask lit_l1
  how
  how 1
  ```

- WHY questions can be used to ask why a question was asked.
  It provides the rule with the asked atom in the body.
  You can ask WHY the rule in the head was asked.

Knowledge-Level Debugging

There are four types of non-syntactic errors that can arise in rule-based systems:

- An incorrect answer is produced: an atom that is false in the intended interpretation was derived.

Knowledge-Level Debugging

There are four types of non-syntactic errors that can arise in rule-based systems:

- An incorrect answer is produced: an atom that is false in the intended interpretation was derived.
- Some answer wasn't produced: the proof failed when it should have succeeded. Some particular true atom wasn't derived.

Knowledge-Level Debugging

There are four types of non-syntactic errors that can arise in rule-based systems:

- An incorrect answer is produced: an atom that is false in the intended interpretation was derived.
- Some answer wasn't produced: the proof failed when it should have succeeded. Some particular true atom wasn't derived.
- The program gets into an infinite loop.

Knowledge-Level Debugging

There are four types of non-syntactic errors that can arise in rule-based systems:

- An incorrect answer is produced: an atom that is false in the intended interpretation was derived.
- Some answer wasn't produced: the proof failed when it should have succeeded. Some particular true atom wasn't derived.
- The program gets into an infinite loop.
- The system asks irrelevant questions.

Debugging incorrect answers (false positives)

- Suppose atom g was proved but is false in the intended interpretation.

Debugging incorrect answers (false positives)

- Suppose atom g was proved but is false in the intended interpretation.
- There must be a rule $g \leftarrow a_1 \wedge \ldots \wedge a_k$ in the knowledge base that was used to prove g.

Debugging incorrect answers (false positives)

- Suppose atom g was proved but is false in the intended interpretation.
- There must be a rule $g \leftarrow a_1 \wedge \ldots \wedge a_k$ in the knowledge base that was used to prove g.
- Either:

Debugging incorrect answers (false positives)

- Suppose atom g was proved but is false in the intended interpretation.
- There must be a rule $g \leftarrow a_1 \wedge \ldots \wedge a_k$ in the knowledge base that was used to prove g.
- Either:
  - one of the a_i is false in the intended interpretation or

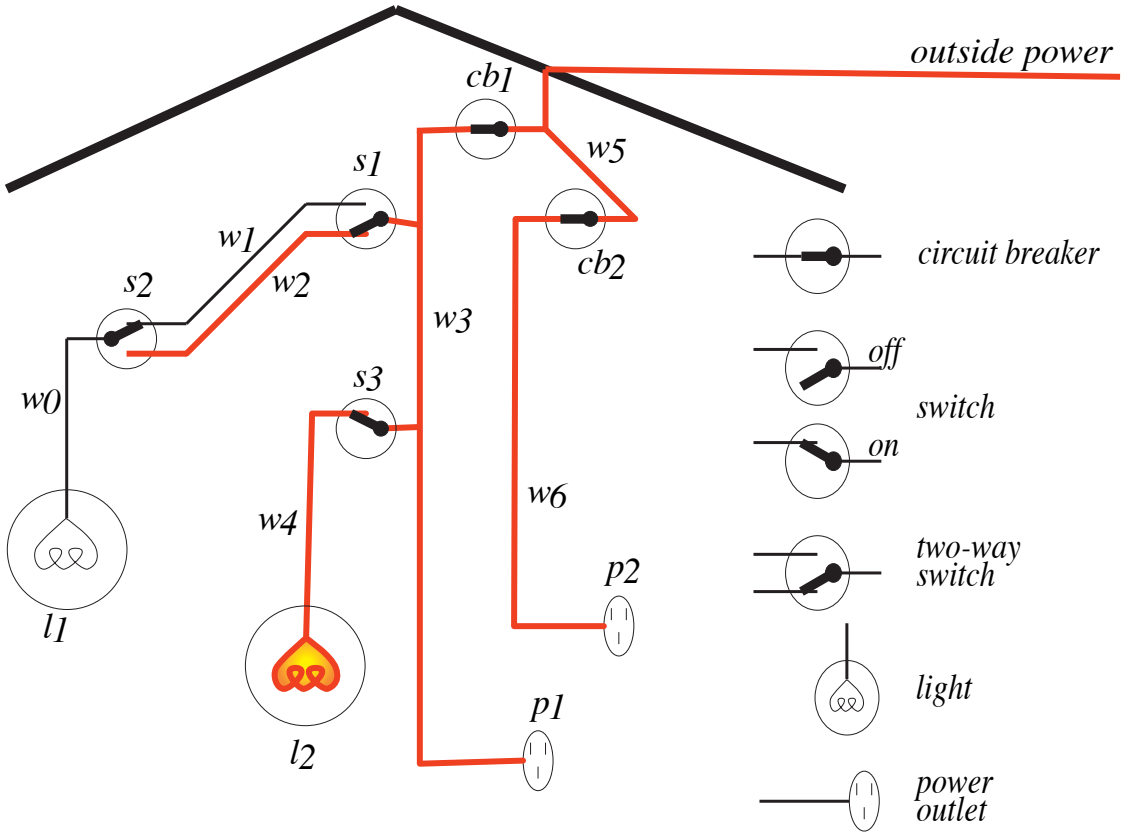
Debugging incorrect answers (false positives)

- Suppose atom g was proved but is false in the intended interpretation.
- There must be a rule $g \leftarrow a_1 \wedge \ldots \wedge a_k$ in the knowledge base that was used to prove g.
- Either:
  - one of the a_i is false in the intended interpretation or
  - all of the a_i are true in the intended interpretation.

Debugging incorrect answers (false positives)

- Suppose atom g was proved but is false in the intended interpretation.
- There must be a rule $g \leftarrow a_1 \wedge \ldots \wedge a_k$ in the knowledge base that was used to prove g.
- Either:
  - one of the a_i is false in the intended interpretation or
  - all of the a_i are true in the intended interpretation.
- Incorrect answers can be debugged by only answering yes/no questions.

Electrical Environment (aipython.org interaction)




```
python -i logicExplain.py
interact(elect_bug)
ask lit_l1
how
how 1
```

Missing Answers (false negatives)

If atom g is true in the intended interpretation, but could not be proved, either:

- There is no appropriate rule for g.

Missing Answers (false negatives)

If atom g is true in the intended interpretation, but could not be proved, either:

- There is no appropriate rule for g.
- There is a rule $g \leftarrow a_1 \wedge \ldots \wedge a_k$ that should have succeeded.

Missing Answers (false negatives)

If atom g is true in the intended interpretation, but could not be proved, either:

- There is no appropriate rule for g.
- There is a rule $g \leftarrow a_1 \wedge \ldots \wedge a_k$ that should have succeeded.
  - One of the a_i is true in the interpretation and could not be proved.