Situated learning approach to design using SOAR

Gourabmoy Nath
Key Centre of Design Computing & Cognition
University of Sydney

Summary

- **Situated learning is an emerging idea in design and is influenced by its parallels in educational instruction and situated cognition.**

- **Present key aspects of situated learning in design.**

- **Model situated learning in design using chunking in the context of an architectural design problem.**

- **Interpret the results of chunking in design in terms of the key aspects of situated learning.**

Situated Learning in design

- **Context plays a key role in shaping the final design artifact and the processes that are used to create the artifact.**
- **Motivation: What design move to make in response to a given context is not well-formalized and tacit. Designing is abductive in nature.**
- **Learning within the authentic context of designing by experimenting within design contexts**
- **Correlating design knowledge to the patterns of the design context under which that knowledge was used.**
- **Correlation = When to apply design knowledge ? How to apply design knowledge ?**
- **Using the learnt knowledge to solve same/similar problems.**

Situated learning $\longrightarrow$ chunking: abstract mapping

- **Learning while doing $\longrightarrow$ chunking occurs while problem-solving.**
- **Experimentation $\longrightarrow$ Search in subgoals**
- **Patterns of context $\longrightarrow$ working memory elements of pre-impasse state relevant to resolving impasse.**
- **When to apply some knowledge $\longrightarrow$ conditions of the chunk**
- **How to apply some knowledge$\longrightarrow$ actions of the chunk**
- **Applicability $\longrightarrow$ chunks integrated in long-term agent memory.**
- **Usefulness $\longrightarrow$ abductive knowledge, $>$ er performance**

Function, behaviour, structure aspect of context

- Design Specifications
  - Lounge: function = allow-living, requires nice-view, adjacent to all other rooms.

```
 (<spec> ^room <r1> + &, <r2> + &
              <r3> + &, <r4> + &
       ^adjacent <a1> + & , <a2> + &,
                 <a3> + &, <a4> + &)
(<r2> ^function allow-living
      ^name lounge)
(<a1> ^room1 <r2> ^room2 <r1>)}
```

  - Bedroom: function= allow-sleeping, requires nice-view
  - Services (Kitchen and Toilet) : function = allow-sleeping
  - Entrance-lobby : function = allow-entry
- All rooms to avoid noise and dust.

Exogenous aspect of context

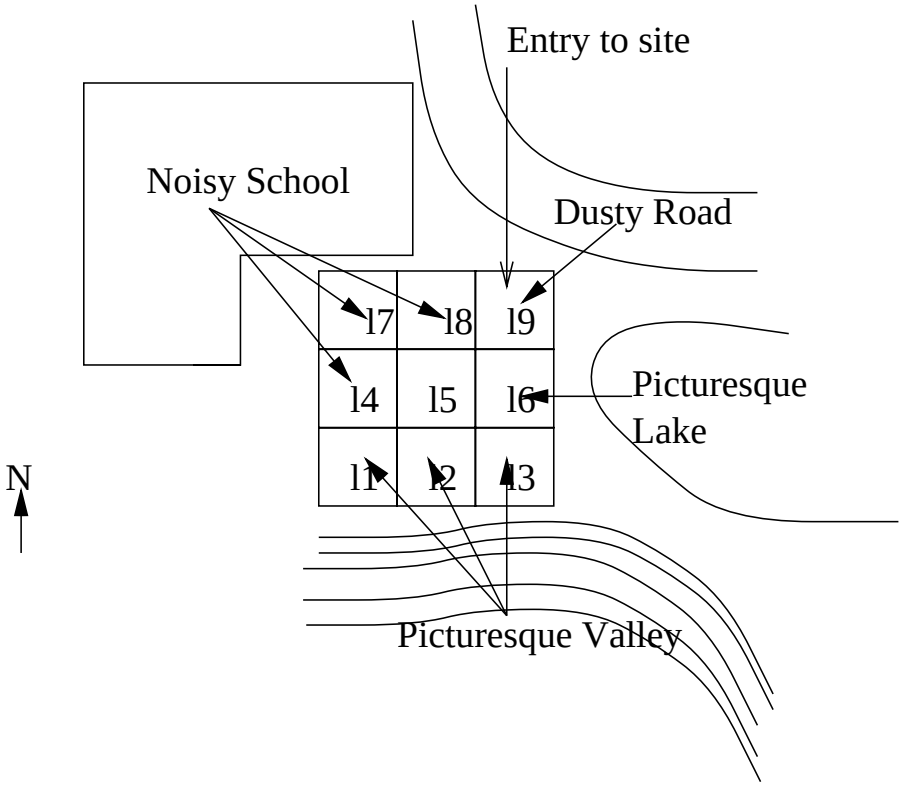
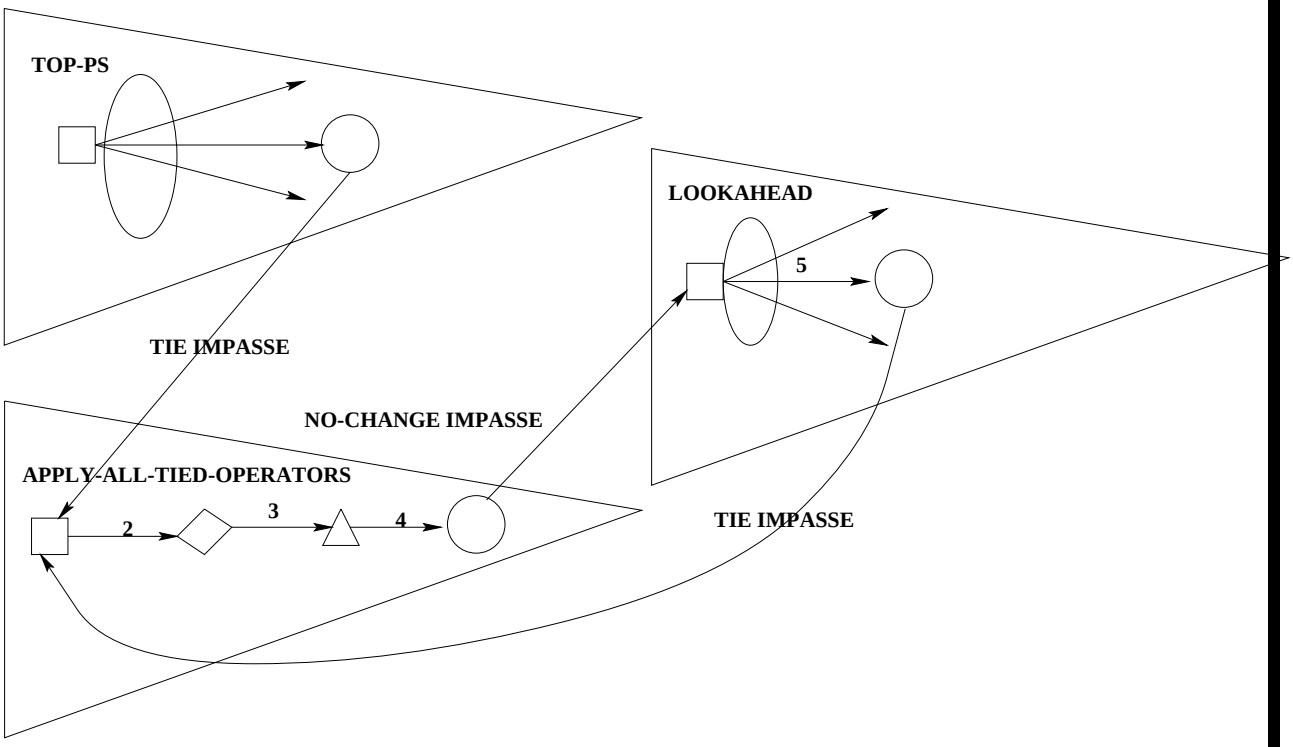


Figure 1: **Site plan for proposed building**

- ```
  (<site> ^zone <z1> + &, <z2> + &,
   <z3> + &, <z4> + &, <z5> + &, <z6> + &
   <z7> + &, <z8> + &, <z9> + &)
   (<z1> ^name l1 ^north <z4> ^east <z2>
         ^abuts <v>)
   (<v> ^is picturesque ^name valley)
   (<z4> ^name l4 ^north <z7> ^east <z5>
         ^abuts <sch>)
   (<v> ^is noisy ^name valley)
  ```
  ```

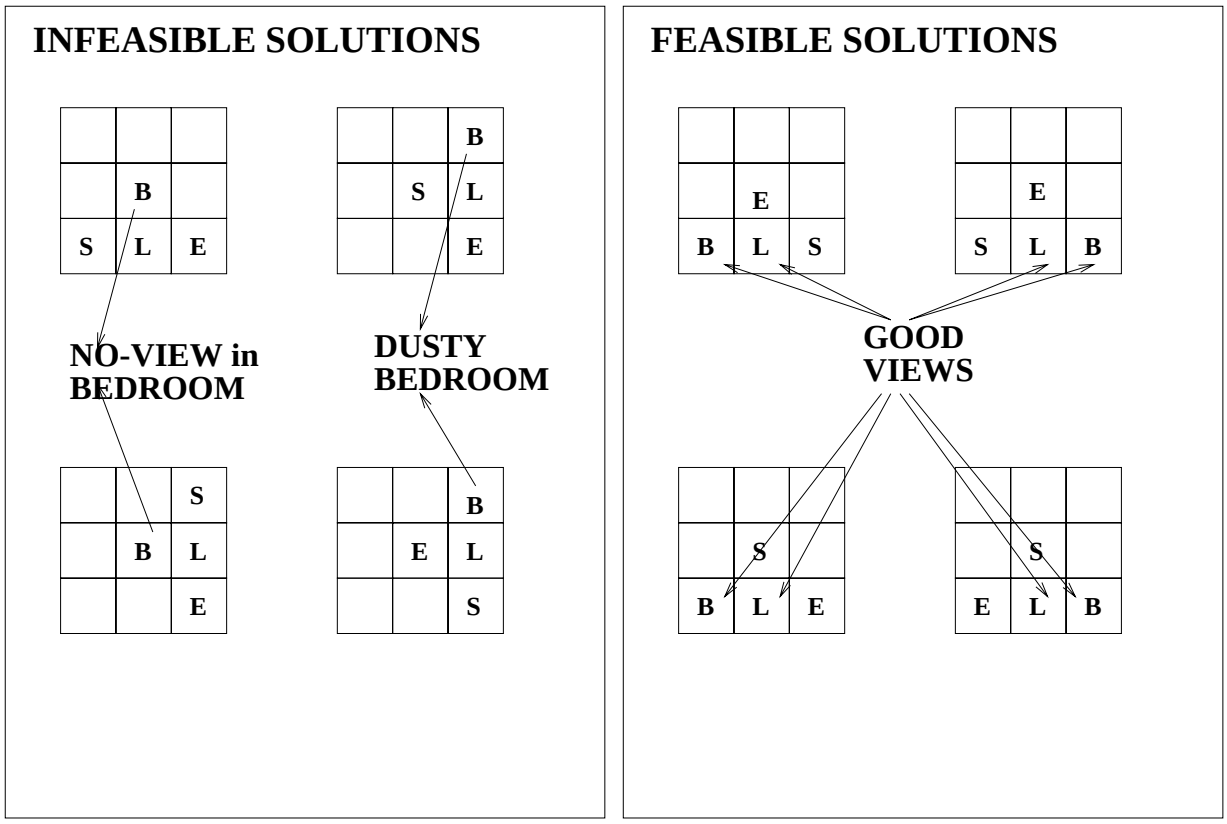
Generative and evaluative design process knowledge



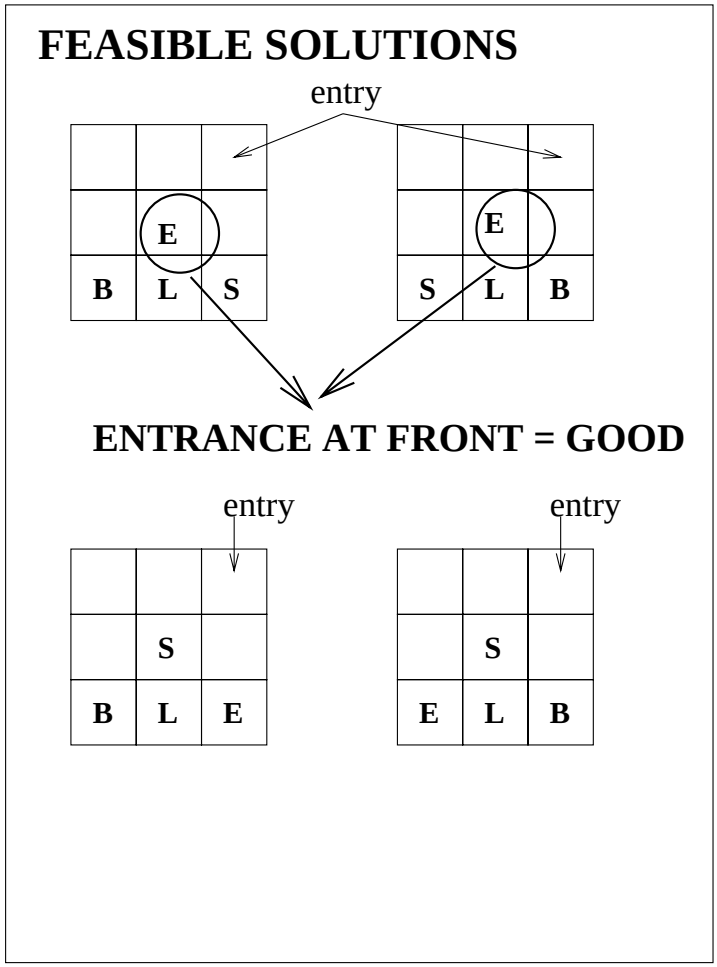
- TOP-PS
  - Propose operator: Place-lounge. 4 instantiations. tie impasse
  - TIE-SPACE = apply-all-tied-operators.
    - For all items in tie-space propose operator to apply $\hat{\text{item}}$
    - Reject operators that have infeasible adjacencies or zones

* Apply operators with feasible designs.
* If all operators are applied and all rooms are not tried
  - State no change impasse = LOOKAHEAD.
  - In problem-space LOOKAHEAD propose operators for the nextuntried room
  - Go to TIE-SPACE
* else
* If all rooms are tried propose operator evaluate-alternatives
* Evaluate-alternatives evaluates a global property of the design.
  - Evaluation = Good if entrance is at Front
* If good design then return to superstate best-preference on parent operator.
* Propagate best preference upwards by recursively returning results ito superstate.

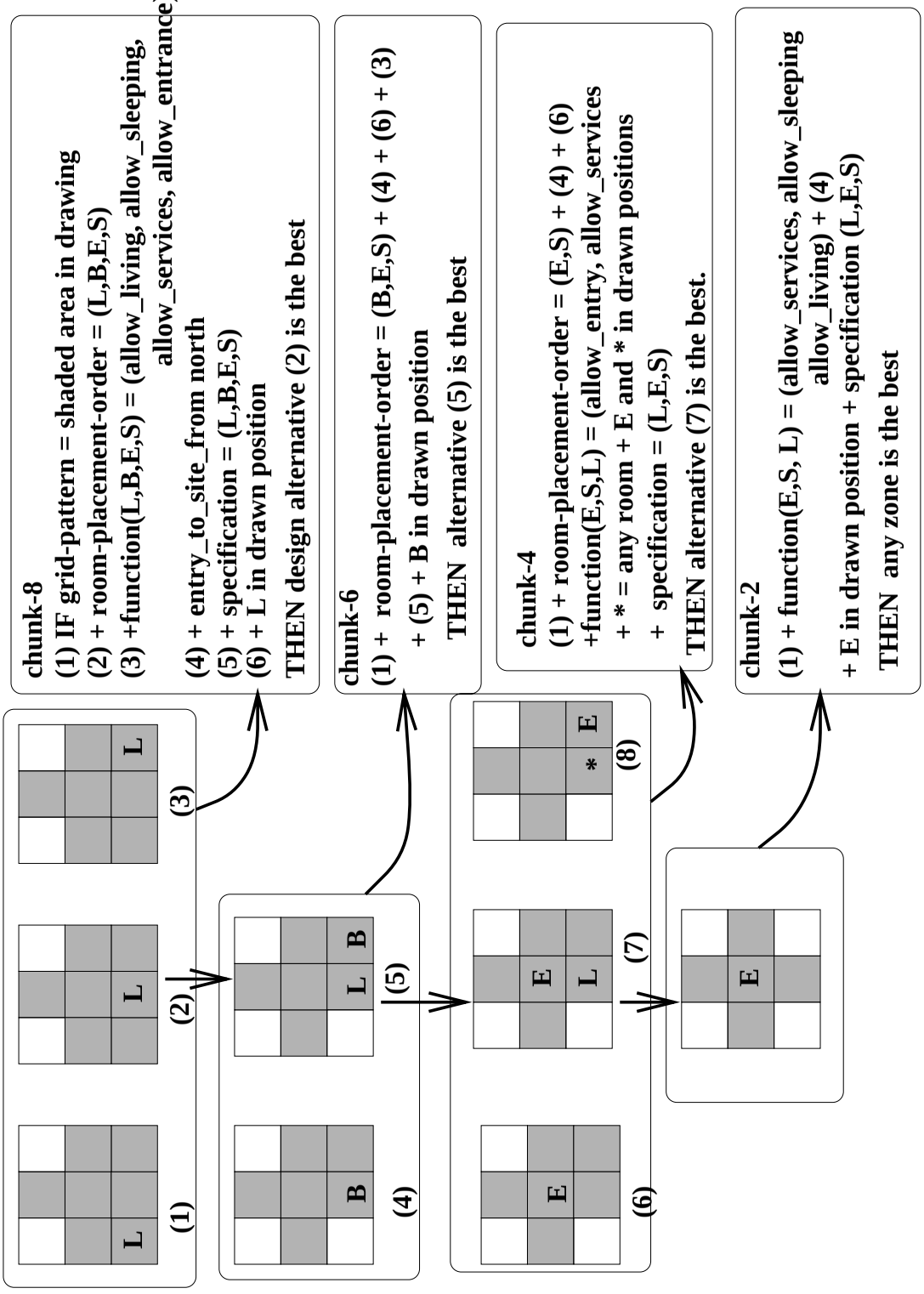
Feasible and Infeasible solutions



Good Solutions among feasible solutions



Learning Results



An example chunk

```
soar> p chunk-8
sp {chunk-8
    :chunk
    (state <s1> ^problem-space <p1> ^site <s2> ^current-room <c1>
          ^specification <s3> ^placement-order <p2> ^placement-order <p3>
          ^placement-order <p4> ^operator <o1> + ^operator <o2> +
          ^operator <o3> + ^placement-order <p5> ^design <d1>)
    (<p1> ^name top-ps)
    (<s2> ^entry-to-site-from north ^zone <z1> ^zone <z2> ^zone <w1>
          ^zone <n1> ^zone <z3> ^zone <w2> ^zone <n2>)
    (<c1> ^function allow-living)
    (<s3> ^room <c1> ^room <r1> ^room <r2> ^room <r3>)
    (<r1> ^function allow-services)
    (<r2> ^function allow-sleeping)
    (<p2> ^room <r1>)
    (<p3> ^next <p2> ^room <r3>)
    (<p4> ^room <r2> ^next <p3>)
    (<o1> ^room <c1> ^zone <z1>)
    (<o2> ^room <c1> ^zone <z2>)
    (<o3> ^room <c1> ^zone <w1>)
    (<r3> ^function allow-entry)
    (<p5> ^next <p4> ^room <c1>)
    (<z1> ^west <w1>)
    (<w1> ^west <z2> ^north <n1>)
    (<z2> ^east <w1>)
    (<n1> ^south <w1> ^west <w2> ^north <n2>)
    (<z3> ^west <n1>)
    -->
    (<s1> ^operator <o3> = ^operator <o3> >)
}
```

Gold

- **Situated learning modelled through chunking encapsulate what design move to make in a given context.**

- **Influencing context elements = LHS of chunk = When to apply design knowledge, How to apply design knowledge = RHS.**

- **Generative and evaluative design knowledge has been converted to abductive design knowledge which matches the notion of design as abduction.**

- **Simple demonstration of situated learning by chunking in architectural design.**

Coal

- Toy problem: Real design problems need solution space sampling
- Intermediate results: Utility of chunks yet to be tested for similar problems
- More tuning of representation needed