PSVN

Robert Holte
Computing Science Department
University of Alberta

What is PSVN?

1. A **declarative language** for defining a state space
2. An **API** that includes type definitions, constants, and functions for manipulating states.
3. A **compiler** (psvn2c) that creates C code implementing the PSVN API given a PSVN state space definition
4. A **suite of software** that uses the PSVN API or supports the use of PSVN

The API Manual & Appendix

- The API Manual defines the types, constants, functions, etc. that any implementation must provide and gives specific guidelines about what may vary from one implementation to another.
- The Appendix gives specific details of the various implementations.

Iterating Through a State's Children

```
state_t state, child ;
ruleid_iterator_t iter ;
int ruleid ;


init_fwd_iter(&iter,&state);
while((ruleid=next_ruleid(&iter)) >= 0){
  apply_fwd_rule(ruleid,&state,&child);
      ...
}
```

Some Code Using the PSVN API

- A*
- IDA*
- GBFS with the FF heuristic (re-uses the FF heuristic code from Fast Downward)
- Code for generating pattern databases
- Stratified search
- etc.

The PSVN Language

Language Design Objectives

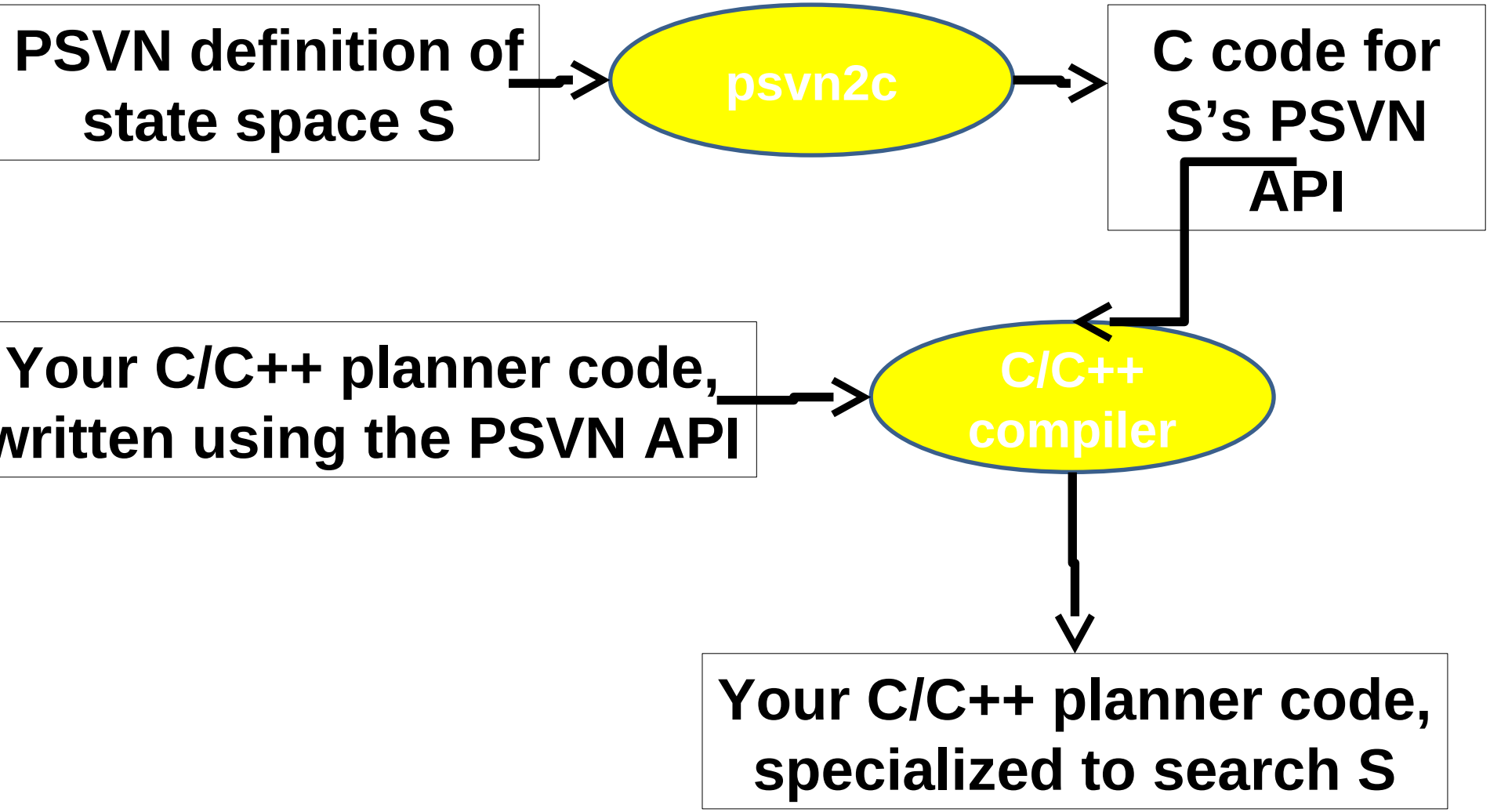
- As simple as possible, but able to represent standard testbeds naturally and compactly.
  - Finite-domain representation (FDR)
  - swapping the values of variables compactly expressed
- Easy to
  - use operators backwards
  - make an abstraction of a state space
  - reason “symbolically” (i.e. about partially specified states)
- Very efficient code can be generated easily from the declarative form.

Production System, Vector Notation

- State = vector of length N.
  - Each entry of the vector is called a state variable.
  - Each state variable has a finite domain of possible values.
- Each operator is of the form LHS => RHS.
  - LHS is the operator's precondition
  - RHS is the operator's effect
  - Both are vectors of length N. Each entry is either:
    - Constant (from the appropriate domain)
    - Variable Symbol (same symbol can occur more than once)

psvn2c, the PSVN Compiler

The Compilation Approach



Value Added by psvn2c

- “Optimizes” the order of precondition tests to quickly find the first rule that applies to a given state.
- Automatic move pruning: can speed up depth-first search by orders of magnitude.
- Infers operator inverses (for backwards search).
- “Optimizes” the number of bits used to represent a state.
- No “interpretation”, highly efficient C code is generated that can then be subjected to all the optimization power of the C compiler.

PSVN Generators

- Most problem domains have parameters. e.g. in the Towers of Hanoi:
  - number of disks
  - number of pegs
- A PSVN generator for a parameterized problem domain is code (in any programming language) that generates PSVN for a specific set of parameter values.
- Like what Fast Downward's translator does with PDDL's static predicates, but PSVN generators can do more:
  - Alternative cost models
  - Alternative encodings of a problem domain

Main Directory

Documents

- PSVN Manual
- PSVN API Manual (with the appendix for the psvn2c implementation)

Lesson01, …, **Lesson05**

- Tutorial Lessons

PSVN Manual

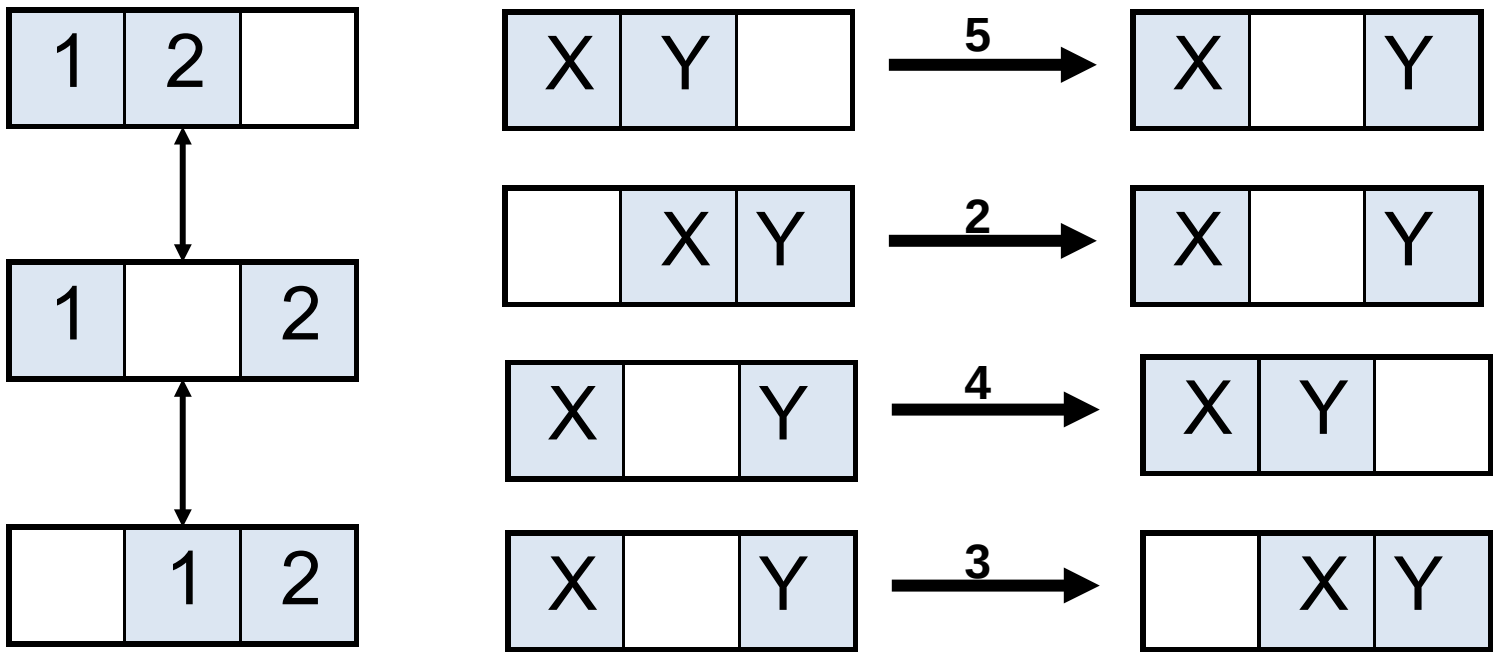
- Complete description of the PSVN language
- Snippets of code illustrating how to use the API
- Overviews of move pruning and abstraction
- **Tutorial lessons** taking you step by step through all of the above and how to use psvn2c

ProblemDomains directory

- PSVN generators for various parameterized problem domains (PPDs).
- Each PPD has a directory of its own with (at least):
  - Code for generating PSVN for a specific version of the PPD, written in some programming language (usually C++)
  - README file explaining what the PPD is, what encodings and cost models the code supports, and how to run the code (command line options, input, etc.)

sliding_tile1x3.psvn

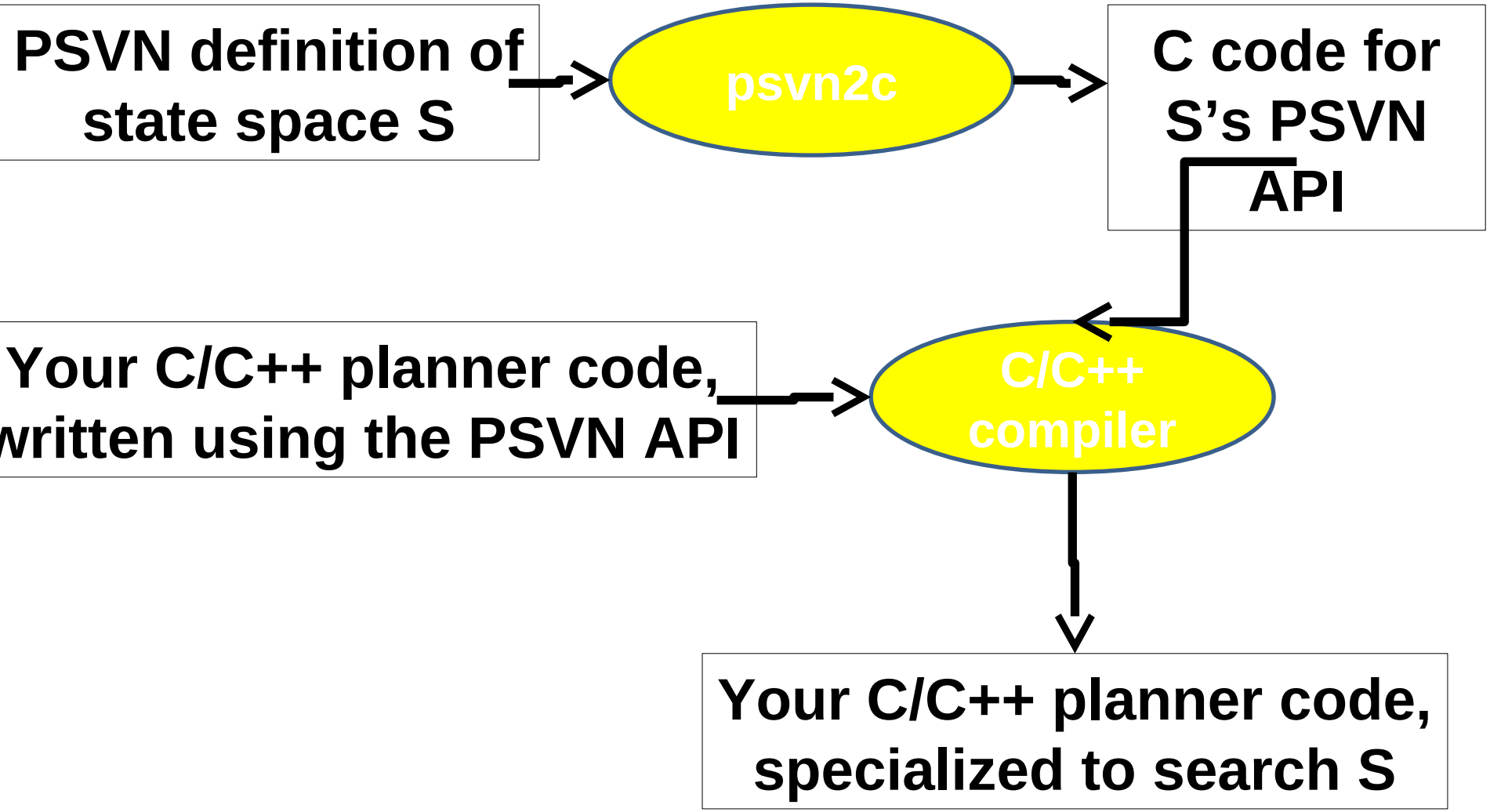
File sliding_tile1x3.psvn contains a PSVN definition for a 1x3 sliding tile puzzle. It uses the symbol b to represent the blank and numbers 1 and 2 to represent the tiles. Each operator has a label and a cost. Although very small, it illustrates almost all the elements of the PSVN language.



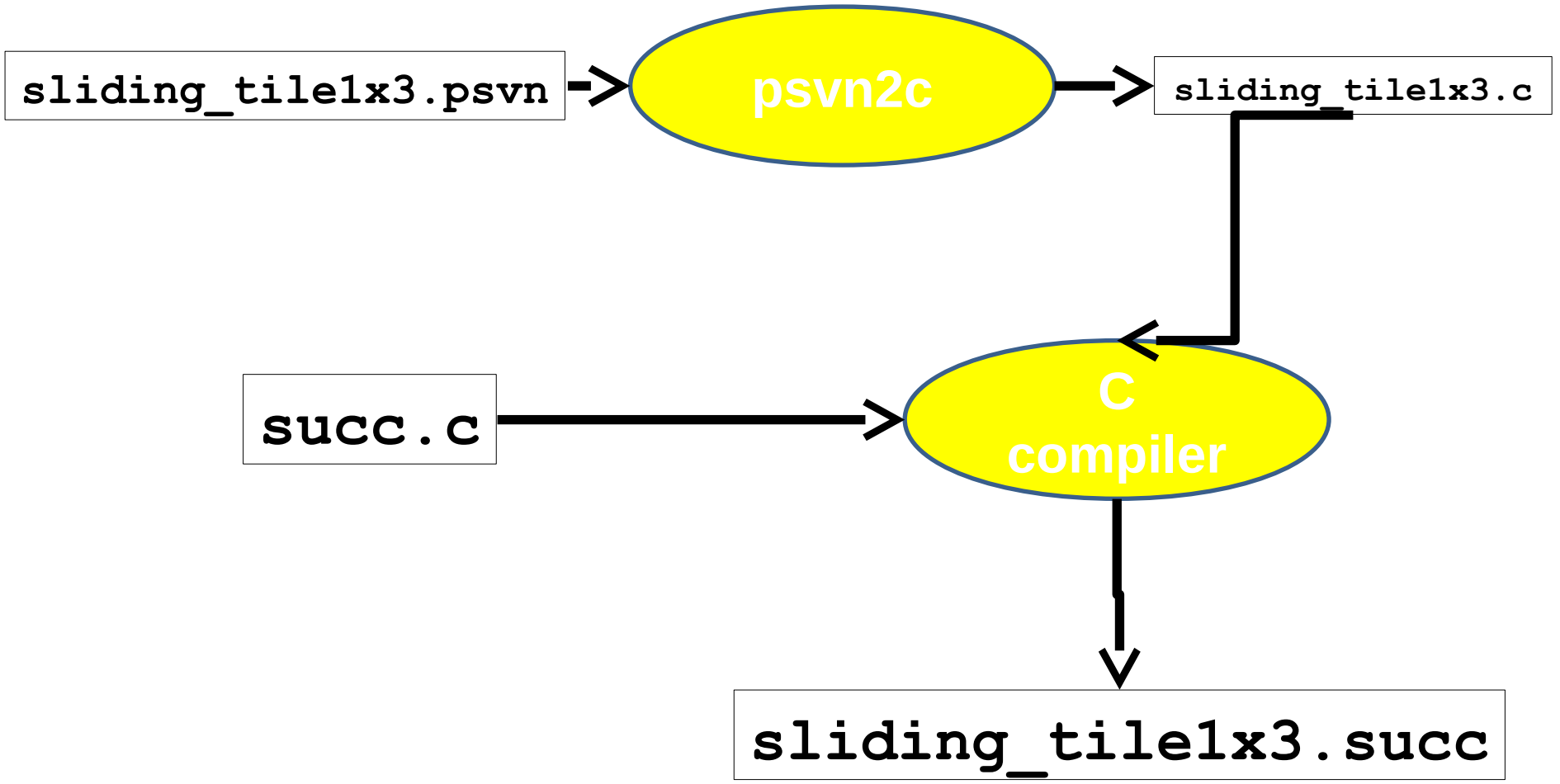
Putting the pieces together

How do we create a version of `succ.c` that is specific to the 1x3 sliding tile puzzle as defined in `sliding_tile1x3.psvn`? The answer is `make sliding_tile1x3.succ.` This will create an executable called `sliding_tile1x3.succ` that will perform the function defined by `succ.c` specifically for the 1x3 sliding tile puzzle as defined in `sliding_tile1x3.psvn`.

The Compilation Approach



What the Makefile Does



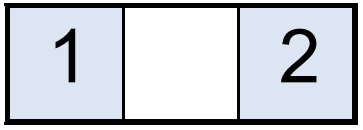
Do It!

```
make sliding_tile1x3.succ
```

Now execute the new file

```
./sliding_tile1x3.succ
```

It asks for input. Enter this state:



Lesson01b – Slide-Jump Puzzle