A Programming Language for Estates and Future Interests

James Grimmelmann
Cornell Tech and Cornell Law School

DIMACS Workshop on Co-Development of Computer Science and Law
May 12, 2022

In this talk

- Motivation
- Inspiration
- Formalization
- Implementation
- Demonstration
- Reflection

Collaborators

- Shrutarshi Basu (Cornell—> Harvard —> Middlebury)
- Nate Foster (Cornell)
- Anshuman Mohan (Cornell)
- Shan Parikh (Cornell)
- Ryan Richardson (Cornell)

Publications

- *Property Conveyances as a Programming Language*, in Proc. 2019 ACM SIGPLAN Int'l Symp. on New Ideas, New Paradigms, & Reflections on Programming and Software (Onward!) 128 (2019)

- *A Programming Language for Estates and Future Interests*, Yale Journal of Law and Technology (forthcoming)

Motivation

Pop Quiz

What interests does the following conveyance create?

"To Sulu for life, then to Uhura and her heirs, but if she marries, then to McCoy and his heirs."

(Peter T. Wendel, *A Possessory Estates and Future Interests Primer* 116 (3d ed. 2007)

Answer

- Sulu has a life estate
- Uhura has a vested remainder in fee simple subject to divestment
- McCoy has a shifting executory interest in fee simple

Estates in Land and Future Interests

- Division of land among multiple owners over time
- Doctrines developed over the past millennium
- Highly formalistic, and extremely rigid

Future interests in legal education

- Future interests were the traditional core of the first-year law-school course in Property
    - Students might spend several months on them
    - Today, a week or two is more common
- Notoriously dry, notoriously incomprehensible
    - There are study aids just for this part of the course!
    - Frequent target of student complaints

Quotes

- “The law is supposed to be a learned profession. Would not that supposition have to be revised if lawyers could no longer talk intelligently about fee tails after possibility of issue extinct, or distinguish a destructible contingent remainder from an executory interest in the nature of a shifting use?” — A. James Casner

- The formulas that govern future interests are similar to those of chemistry. They seem to be more of the law of nature than law of men except for one crucial difference: The rules of future interests occasionally make no sense.” – Daniel B. Bogart

Inspiration

Conveyances are a *programming language*

- Future interests feel different from other parts of law because ambiguity and discretion have been wrung out

- Centuries of judicial rigidity turned future interests into a highly formalized system mostly following consistent rules
    - "to X for life" —> X's interest terminates at X's death
    - "Y, but if C to X" —> X's interest cuts off Y if C occurs

- These rules have the modular recursive generative structure of a programming language

Our project

- Formalize the language of conveyances (Orlando)
  - Syntax
  - Semantics
- Create an interpreter for that language (Littleton)
  - OCaml back-end
  - Command-line and Bootstrap+JS front-ends

Formalization

Moving parts

1. Concrete syntax of ***conveyances***
   Represents the actual language used by a lawyer

2. Abstract data type of ***title trees***
   Represents the legal interests in a piece of property

3. Syntax-directed ***translation function*** from (1) to (2)
   Captures the speech-act effects of legal language

4. ***Semantics*** to update (2) in response to events
   Implements the doctrines of property law

(1) Conveyance syntax is a context-free grammar

$$conveyance \Rightarrow \overbrace{\texttt{Owner}}^{person}\ \texttt{conveys}\ \overbrace{\texttt{to Alice}}^{grant}$$

$$grant \Rightarrow \overbrace{\texttt{to Alice for life}}^{grant}\ \texttt{then}\ \overbrace{\texttt{to Bob}}^{grant}$$

$$grant \Rightarrow \texttt{to}\ \overbrace{\texttt{Alice}}^{person}\ \overbrace{\texttt{for life}}^{quantum}$$

Conveyance syntax grammar

conveyance ⇒ *owner* conveys *grant*

grant ⇒ to *person quantum*
grant ⇒ *grant limitation*
grant ⇒ if *condition grant*
grant ⇒ if *condition grant* otherwise *grant*
grant ⇒ *grant* but if *condition grant*
grant ⇒ *grant* but if *condition* ... reenter
grant ⇒ *grant* then *grant*
grant ⇒ (*grant*)

quantum ⇒ ϵ
quantum ⇒ and *pronoun* heirs
quantum ⇒ and the heirs of *pronoun* body
quantum ⇒ for life
quantum ⇒ for the life of *person*
quantum ⇒ for *n* years

limitation ⇒ while *condition*
limitation ⇒ until *condition*

person ⇒ O|A|B|C| ... |Alice|Bob| ...
pronoun ⇒ her|his|hir|their|zir| ...

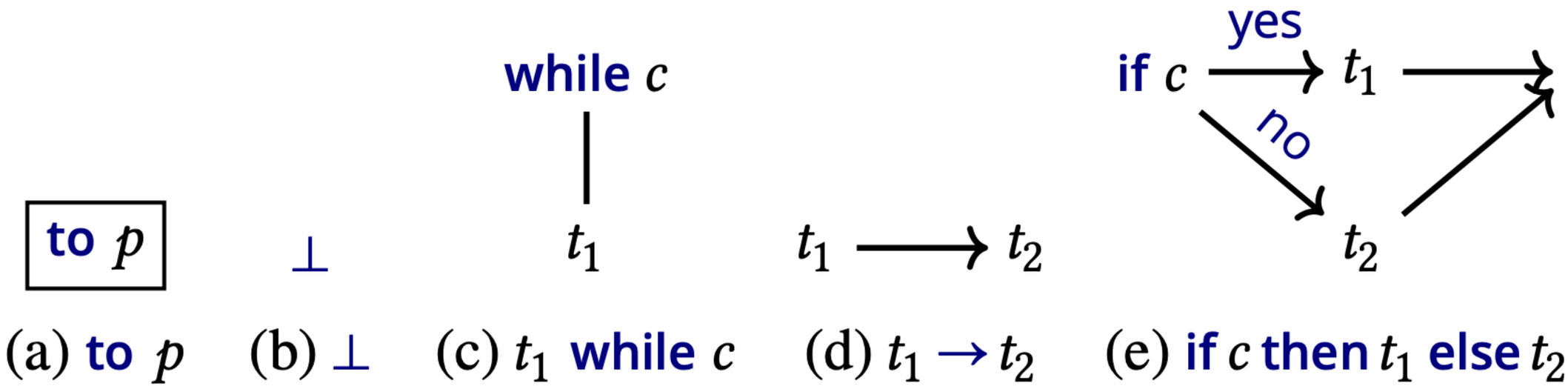
Parsing

conveyance
⇒ *person* conveys *grant*
⇒ Owner conveys *grant*
⇒ Owner conveys *grant* then *grant*
⇒ Owner conveys to *person quantum* then *grant*
⇒ Owner conveys to Alice *quantum* then *grant*
⇒ Owner conveys to Alice for life then *grant*
⇒ Owner conveys to Alice for life then to *person quantum*
⇒ Owner conveys to Alice for life then to Bob *quantum*
⇒ Owner conveys to Alice for life then to Bob and his heirs

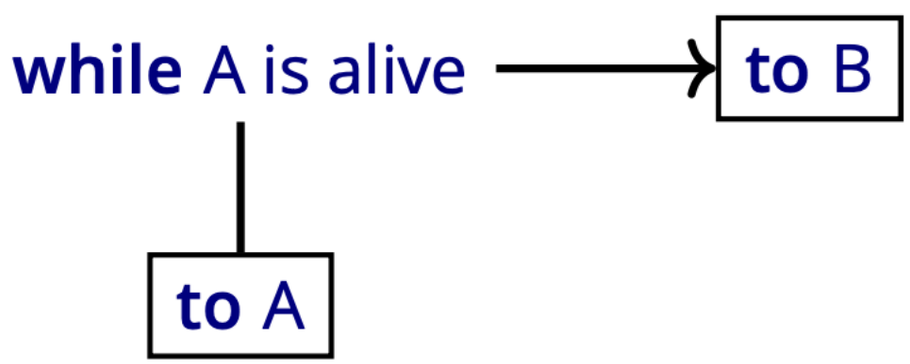
(2) Title trees

$$
\begin{array}{lcl}
t & \Rightarrow & \textbf{to}\ p \\
t & \Rightarrow & \bot \\
t & \Rightarrow & t_1\ \textbf{while}\ c \\
t & \Rightarrow & \textbf{if}\ c\ \textbf{then}\ t_1\ \textbf{else}\ t_2 \\
t & \Rightarrow & t_1 \rightarrow t_2
\end{array}
$$

Graphical representation of title trees



(a) **to** p (b) $\bot$ (c) t_1 **while** c (d) $t_1 \to t_2$ (e) **if** c **then** t_1 **else** t_2



(**to** A **while** A is alive) → **to** B

(3) Translation function

$[\![\textit{owner}\ \texttt{conveys}\ \textit{grant}]\!](t) \;=\; t[([\![\textit{grant}]\!]_{\textit{owner}} \rightarrow \textbf{to}\ \textit{owner})\ /\ \textbf{to}\ \textit{owner}]$

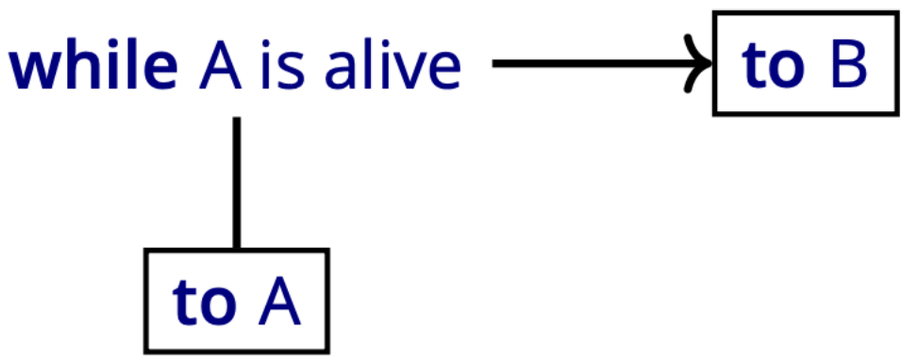
$[\![\texttt{to}\ \textit{person}\ \texttt{true}]\!]_o \;=\; \textbf{to}\ \textit{person}$
$[\![\texttt{to}\ \textit{person}\ \textit{quantum}]\!]_o \;=\; \textbf{to}\ \textit{person}\ \textbf{while}\ [\![\textit{quantum}]\!]_{\textit{person}}$
$[\![\textit{grant}\ \textit{limitation}]\!]_o \;=\; [\![\textit{grant}]\!]_o\ \textbf{while}\ [\![\textit{limitation}]\!]$
$[\![\texttt{if}\ \textit{condition}\ \textit{grant}]\!]_o \;=\; \textbf{if}\ [\![\textit{condition}]\!]\ \textbf{then}\ [\![\textit{grant}]\!]_o\ \textbf{else}\ \bot$
$[\![\texttt{if}\ \textit{condition}\ \textit{grant}_1\ \texttt{otherwise}\ \textit{grant}_2]\!]_o \;=$
$\quad \textbf{if}\ [\![\textit{condition}]\!]\ \textbf{then}\ [\![\textit{grant}_1]\!]_o\ \textbf{else}\ [\![\textit{grant}_2]\!]_o$
$[\![\textit{grant}_1\ \texttt{but if}\ \textit{condition}\ \textit{grant}_2]\!]_o \;=$
$\quad ((([\![\textit{grant}_1]\!]_o \rightarrow \textbf{to}\ p)\ \textbf{while}\ [\![\textit{condition}]\!]) \rightarrow [\![\textit{grant}_2]\!]_o$
$[\![\textit{grant}\ \texttt{but if}\ \textit{condition}\ \ldots\texttt{reenter}]\!]_o \;=$
$\quad ([\![\textit{grant}_1]\!]_o\ \textbf{while}\ [\![\textit{condition}]\!]\ \text{and}\ o\ \text{does not reenter}) \rightarrow \textbf{to}\ o$
$[\![\textit{grant}\ \texttt{then}\ \textit{grant}_2]\!]_o \;=\; [\![\textit{grant}_1]\!]_o \rightarrow [\![\textit{grant}_2]\!]_o$
$[\![(\ \textit{grant}\ )]\!]_o \;=\; [\![\textit{grant}]\!]_o$

$[\![\epsilon]\!]_p \;=\; \text{true}$
$[\![\text{and} \ldots \text{heirs}]\!]_p \;=\; \text{true}$
$[\![\text{and the heirs} \ldots \text{body}]\!]_p \;=\; p\ \text{has issue}$
$[\![\text{for life}]\!]_p \;=\; p\ \text{is alive}$
$[\![\text{for the life of}\ q]\!]_p \;=\; q\ \text{is alive}$
$[\![\text{for}\ n\ \text{years}]\!]_p \;=\; n\ \text{years have not yet passed}$

$[\![\texttt{while}\ \textit{condition}]\!] \;=\; [\![\textit{condition}]\!]$
$[\![\texttt{until}\ \textit{condition}]\!] \;=\; \neg[\![\textit{condition}]\!]$

Translation

$$
\begin{aligned}
& [\![\texttt{to A for life, then to B and her heirs}]\!]_O \\
= \quad & [\![\texttt{to A for life}]\!]_O \to [\![\texttt{to B and her heirs}]\!]_O \\
= \quad & [\![\texttt{to A A is alive}]\!]_O \to [\![\texttt{to B and her heirs}]\!]_O \\
= \quad & (\textbf{to A while A is alive}) \to [\![\texttt{to B and her heirs}]\!]_O \\
= \quad & (\textbf{to A while A is alive}) \to [\![\texttt{to B true}]\!]_O \\
= \quad & (\textbf{to } A \textbf{ while A is alive}) \to \textbf{to B}
\end{aligned}
$$



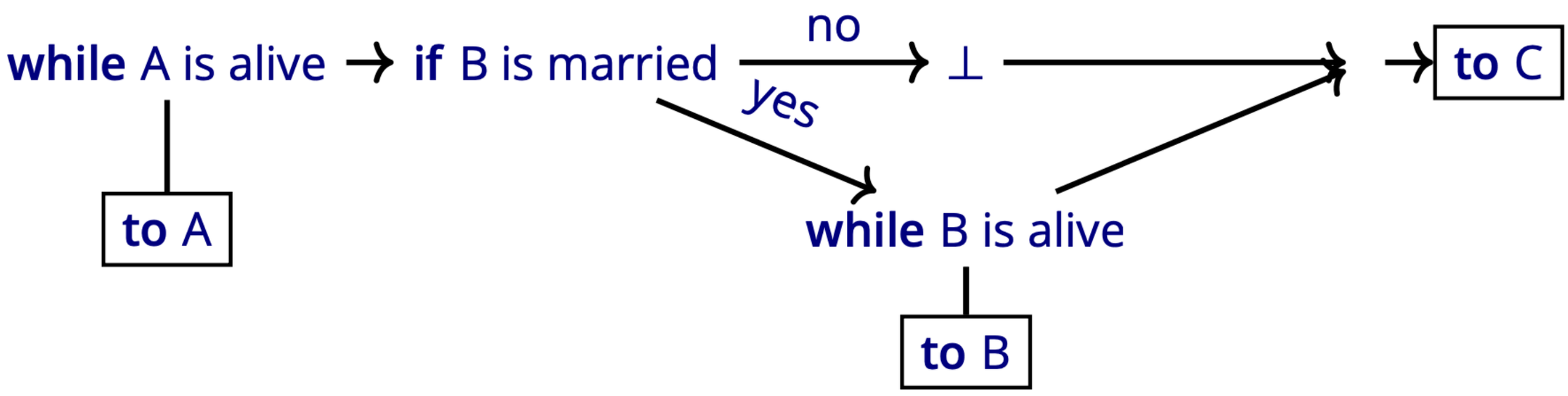
(4) Title tree semantics

$$
\begin{aligned}
\delta(\textbf{to } p) &= \textbf{to } p \\
\delta(\bot) &= \bot \\
\delta(t \textbf{ while } c) &= \begin{cases} \delta(t) \textbf{ while } c & \text{if } \models c \text{ and } \delta(t) \neq \bot \\ \bot & \text{if } \not\models c \\ \bot & \text{if } \delta(t) = \bot \end{cases} \\
\delta(\textbf{if } c \textbf{ then } t_1 \textbf{ else } t_2) &= \begin{cases} \delta(t_1) & \text{if } \models c \\ \delta(t_2) & \text{if } \not\models c \end{cases} \\
\delta(t_1 \to t_2) &= \begin{cases} \delta(t_1) \to t_2 & \text{if } \delta(t_1) \neq \bot \\ \delta(t_2) & \text{if } \delta(t_1) = \bot \end{cases}
\end{aligned}
$$

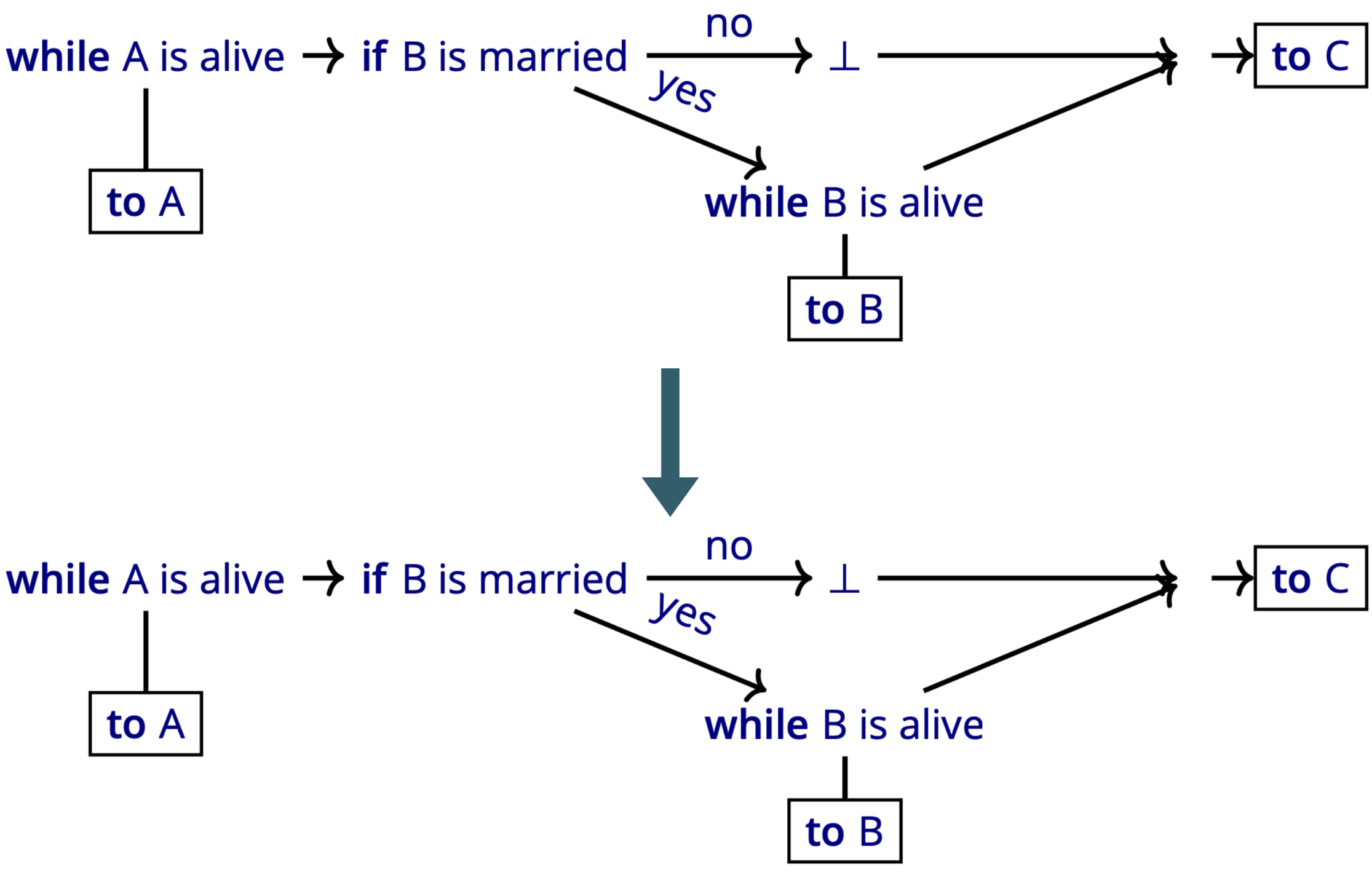
A conveyance and subsequent events

```
O conveys to A for life, then if B is
  married to B for life, then to C.
B marries.
A dies.
B dies.
```

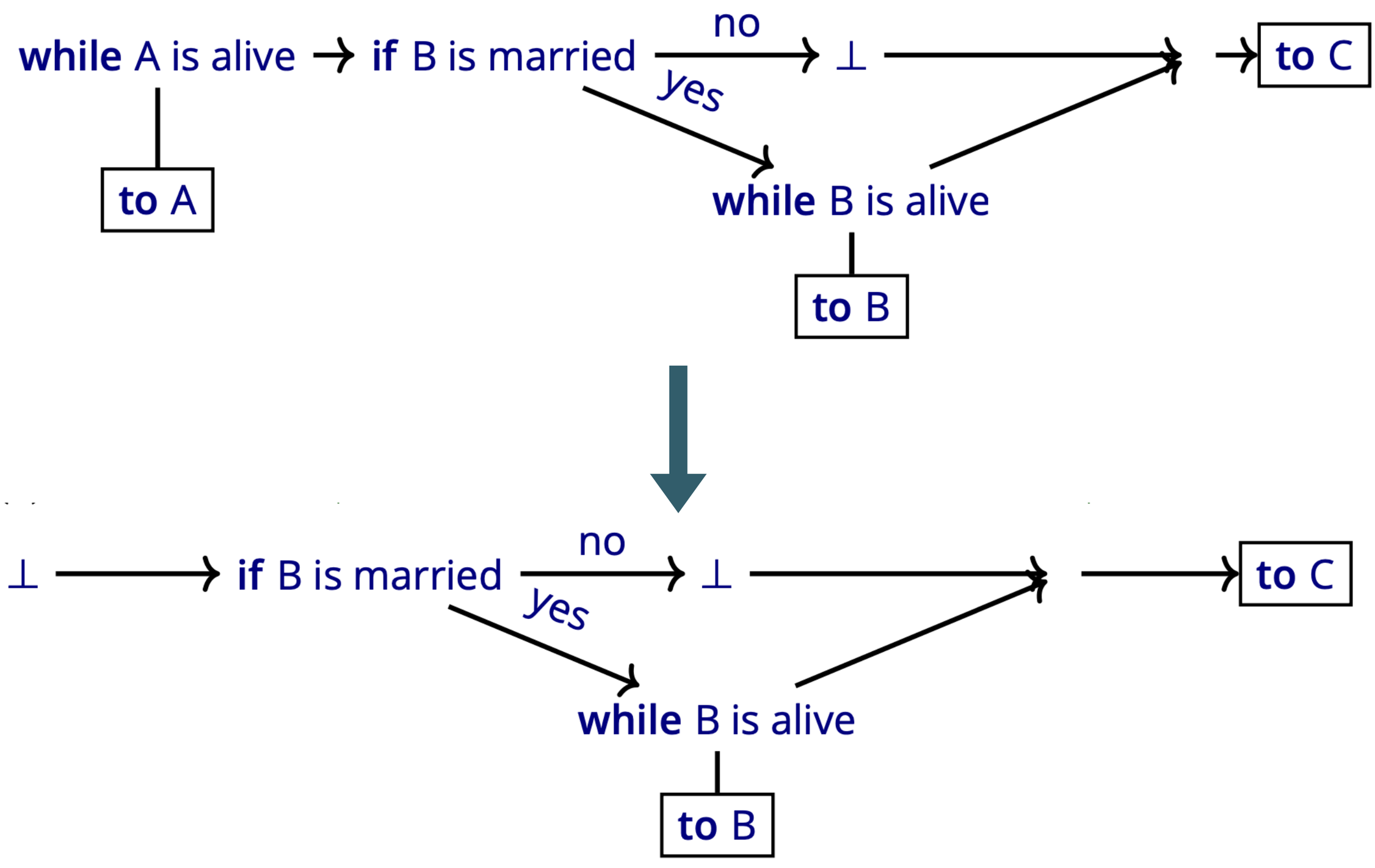
(**to** A **while** A is alive) →
(**if** B is married **then** (**to** B **while** B is alive) **else** ⊥) →
to C



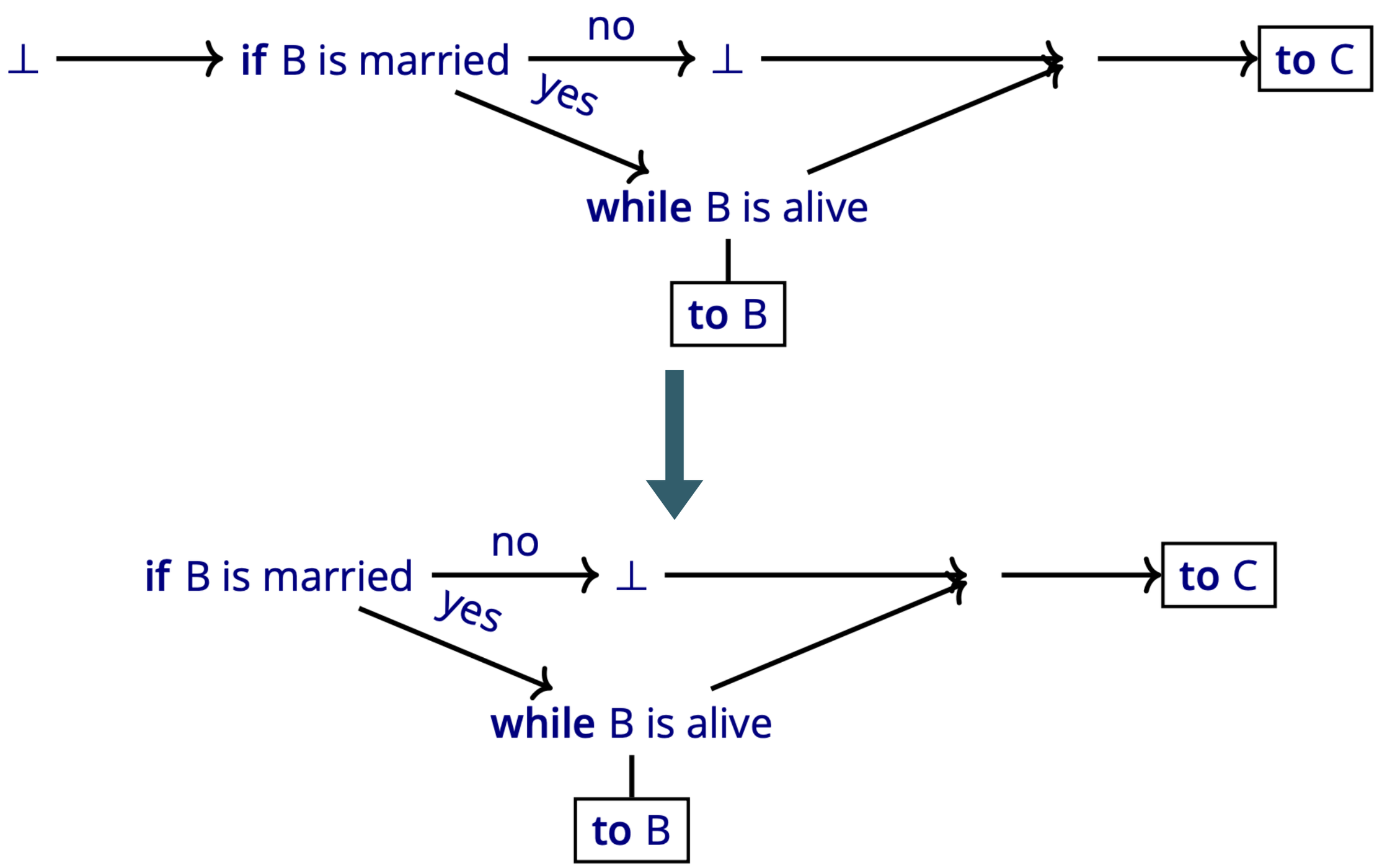
B marries. (1)

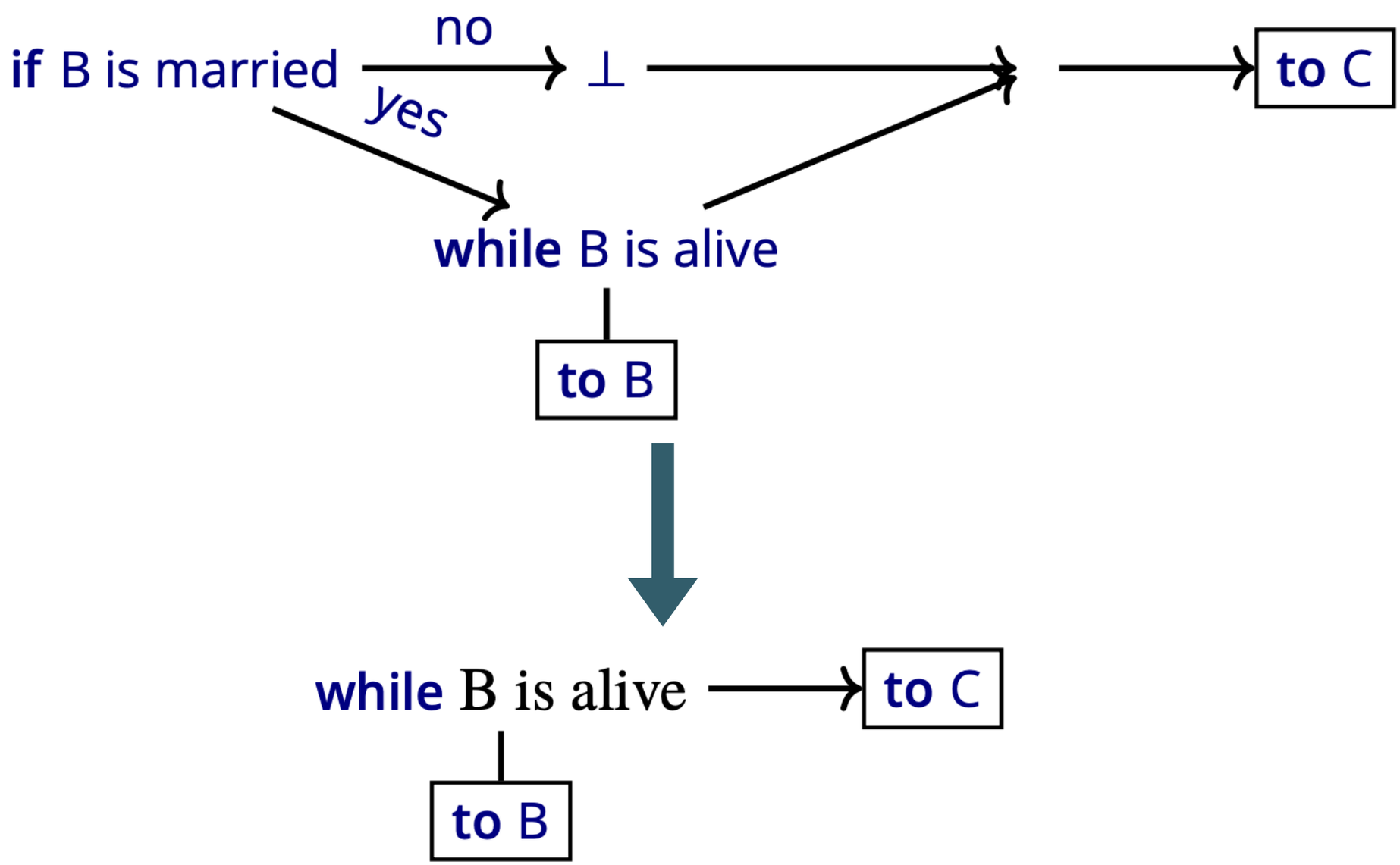

while A is alive
if B is married
no
⊥
to C
yes
to A
while B is alive
to B
while A is alive
if B is married
no
⊥
to C
yes
to A
while B is alive
to B

A dies. (2)

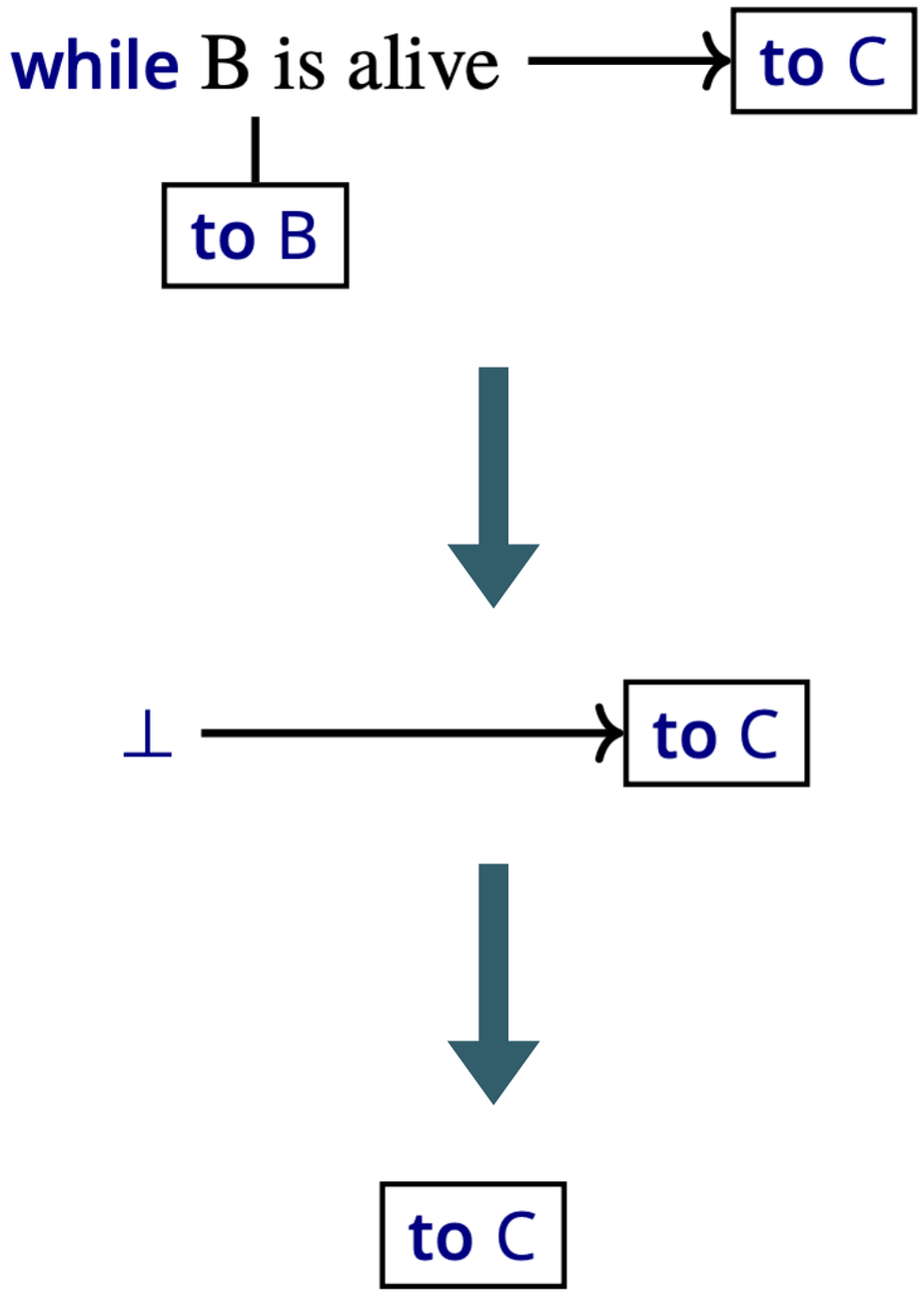

while A is alive
if B is married
no
⊥
to C
yes
to A
while B is alive
to B
⊥
if B is married
no
⊥
to C
yes
while B is alive
to B

(3)


⊥
if B is married
no
yes
⊥
to C
while B is alive
to B
if B is married
no
yes
⊥
to C
while B is alive
to B


if B is married
no
yes
⊥
to C
while B is alive
to B
while B is alive
to C
to B

B dies. (5)



Implementation

Littleton back-end

- ~6000 lines of OCaml
  - ~3000 lines of framework
  - ~1000 lines of legal semantics
  - ~1000 lines of GOFAI to reason about events
  - ~1000 lines of parsing code

Littleton front end

- Command-line “interpreter” for Orlando
- Interactive web app
  - Compiled to JS that runs in-browser
  - Bootstrap interface
  - Available at https://conveyanc.es

Demonstration

Reflection

Advantages of formalization

- Captures the existing rules law students learn
- Recovers the underlying logic that has been obscured by generations of rote learning
- Reveals broader patterns hidden by the proliferation of names and details
- Provides a principled basis for critique and reform

Legal variation = different formal rules

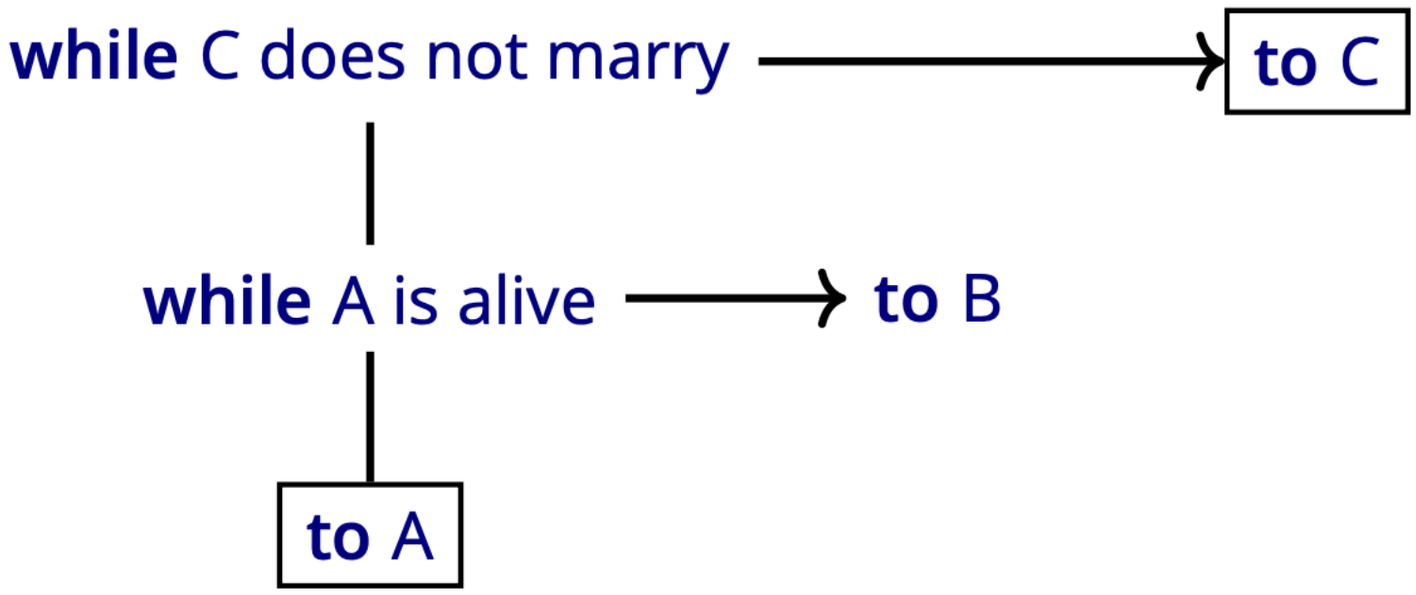
$$
\begin{aligned}
[\![\texttt{and ... heirs}]\!]_p &= \text{true} \\
[\![\epsilon]\!]_p &= p \text{ is alive} \\
[\![\texttt{for life}]\!]_p &= p \text{ is alive} \\
[\![\texttt{for the life of } q]\!]_p &= q \text{ is alive}
\end{aligned}
$$

(a) Default quantum (older rule)

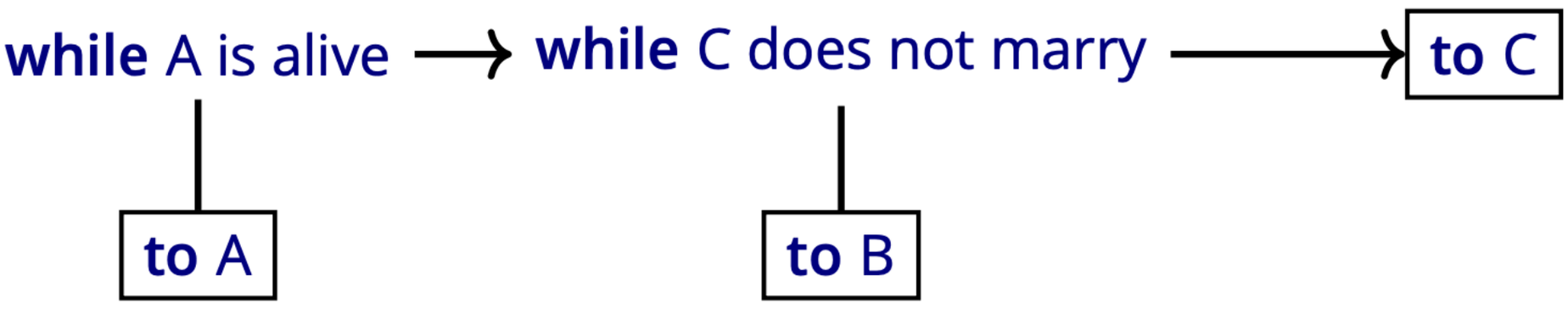
$$
\begin{aligned}
[\![\texttt{and ... heirs}]\!]_p &= \text{true} \\
[\![\epsilon]\!]_p &= \text{true} \\
[\![\texttt{for life}]\!]_p &= p \text{ is alive} \\
[\![\texttt{for the life of } q]\!]_p &= q \text{ is alive}
\end{aligned}
$$

(b) Default quantum (modern rule)

Syntactic ambiguity

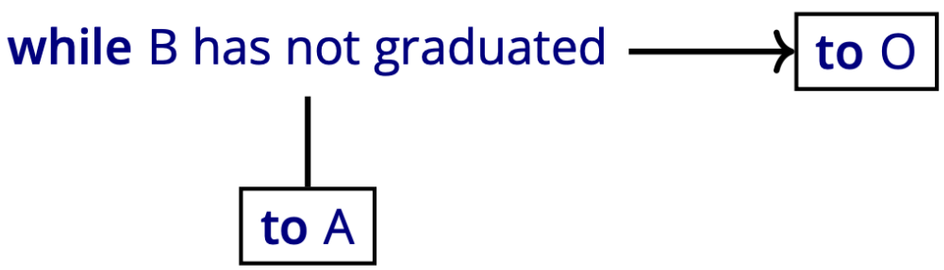


(a) (to A for life, then to B), but if C marries to C

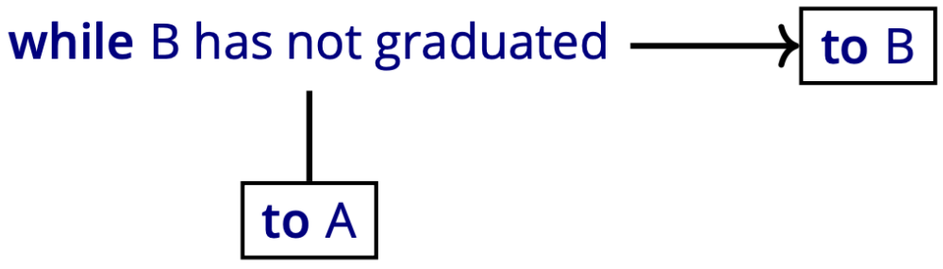


(b) to A for life, then (to B, but if C marries to C)

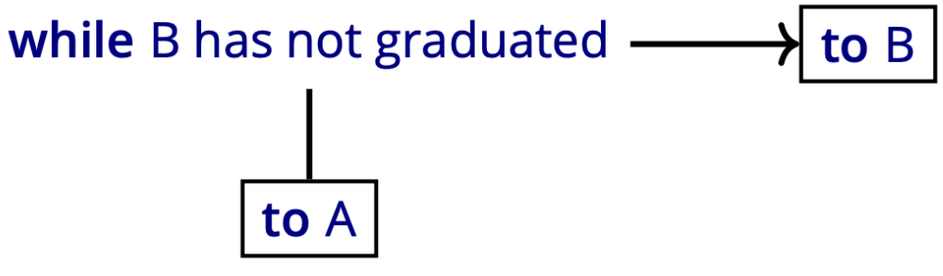
Unnecessary complexity



(a) `to A until B graduates college, then to O` gives A a fee simple determinable



(b) `to A until B graduates college, then to B` gives A a fee simple with executory limitation



(c) `to A, but if B graduates college to B` gives A a fee simple subject to executory limitation

Advantages of implementation

- Visualization for education
- Interactivity for experimentation

A programming-language approach to law

- Treating future interests as a programming language captures the *linguistic structure* of conveyances

- Fits legal areas with rule-driven linguistic structure

- Other suitable bodies of law: property, contracts, tax

Questions