Little (but Hard) Theorems About Big Systems: Some Case Studies

J Strother Moore
Department of Computer Sciences
University of Texas at Austin

Marktoberdorf Summer School 2004

Lecture 3

Definitions

When using a formal system it is necessary to be able to *extend* the theory with the introduction of new concepts.

For example, if we cannot introduce the new functions `rev` or `rev1`, how can we prove anything about them?

But it is risky to add new axioms to a formal system in an undisciplined way.

How do we know the system is consistent after adding a new axiom?

A *Definitional Principle* permits one to extend a formal system without risk.

Theorem. `(p x)` $\rightarrow$ `(q x)`.

Proof:

Definition: `(defun r (x) ...)`

Lemma 1: `(p x)` $\rightarrow$ `(r x)`.

Lemma 2: `(r x)` $\rightarrow$ `(q x)`.

Q.E.D.

But how do you know `(p x)` $\rightarrow$ `(q x)` is a theorem in the original theory? You added an axiom!

Definitions

$\texttt{(defun}\ f\ \ (v_1\ \ \ldots\ \ v_n)\ \ \beta\texttt{)}$

introduces the new axiom

$\forall v_1 \ldots v_n\ \texttt{:}\ \ (f\ \ v_1 \ldots v_n)\ \texttt{=}\ \beta$

provided the definition is *admissible*.

Such definitions are *conservative*: the only new theorems are ones involving the new symbol.

(`defun` f (v_1 ... v_n) β)

is admissible iff

1. f is a new function symbol
2. the v_i are distinct variable symbols
3. β is a term that contains no (free) variable symbols other than the v_i

```
(defun car (x) x)   ; Violates 1

(defun f (x x) x)   ; Violates 2

(defun g (x) y)     ; Violates 3
```

Theorem: $2=7$

Proof:

We have the axiom $\forall$`x: (g x)=y`.

$(\mathtt{g}\ \mathtt{0}) = \mathtt{2}$

$(\mathtt{g}\ \mathtt{0}) = \mathtt{7}$

$\mathtt{2} = \mathtt{7}$

Q.E.D.

What is the harm in non-terminating functions?

```
(defun f (x)
  (if (equal x 1)
      nil
      (cons nil (f (- x 1)))))
```

Lemma:

$\texttt{(natp n)} \wedge \texttt{x} < \texttt{1} \rightarrow \texttt{n} < \texttt{(len (f x))}.$

Theorem: `(len (f 0)) < (len (f 0))`.

Admissiblity (continued)

4. There is a natural number measure ($\mathtt{m}\ v_1 \ldots v_n$) such that for every recursive call ($\mathtt{f}\ \delta_1 \ldots \delta_n$) in β and its governing tests τ:

Measure Theorem:

$$
\begin{array}{l}
(\mathtt{implies}\ \tau \\
\qquad\qquad (\mathtt{<}\ (\mathtt{m}\ \delta_1\ \ldots\ \delta_n) \\
\qquad\qquad\qquad (\mathtt{m}\ v_1\ \ldots\ v_n)))
\end{array}
$$

```
(defun app (x y)
  (if (consp x)
      (cons (car x)
            (app (cdr x) y))
     y))
```

`(consp x)` governs `(app (cdr x) y)`.

Measure Theorem:

```
(implies (consp x)
         (< (m (cdr x) y)
            (m x y)))
```

Measure Theorem:

```
(implies (consp x)
         (< (m (cdr x) y) (m x y)))
```

where we measure the length of `x`

```
(defun m (x y) (len x))
```

or the "tree size"

```
(defun m (x y) (acl2-count x))
```

Elaboration: `m` may return an ordinal, in which case, < should be replaced by `o<`.

Note: ACL2 represents the ordinals up to ϵ_0.

$$\epsilon_0 = \omega^{\omega^{\omega^{\cdots}}}$$

Examples (ordered by o<)

0	0
1	1
2	2
ω	((1 . 1) . 0)
$\omega \times 2$	((1 . 2) . 0)
$\omega \times 2 + 23$	((1 . 2) . 23)
$\omega^2 \times 3 + \omega \times 7 + 19$	((2 . 3) (1 . 7) . 19)
ω^ω	((((1 . 1) . 0) . 1) . 0)

ACL2 provides support for ordinal arithmetic.

$$\omega^2 \times 3 + \omega \times 7 + 19$$

```
(o+ (o* (o^ (omega) 2) 3)
    (o* (omega) 7)
    19)
=
((2 . 3) (1 . 7) . 19)
```

Induction

To prove `(`ψ `x y)` by induction on `x` prove:

Base Case:

```
(implies (not (consp x)) (ψ x y))
```

Induction Step:

```
(implies (and (consp x)
              (ψ (car x) α1)
              (ψ (cdr x) α2))
         (ψ x y))
```

where the α_i are arbitrary terms.

Elaborations

You may, of course, omit an induction hypothesis.

Induction Step:

$$
\begin{array}{l}
\texttt{(implies (and (consp x)} \\
\qquad\qquad\qquad\quad \texttt{(}\psi\ \texttt{(cdr x)}\ \alpha_2\texttt{))} \\
\qquad\qquad\quad \texttt{(}\psi\ \texttt{x y))}
\end{array}
$$

You may have multiple α_i.

Induction Step:

```
(implies (and (consp x)
              (ψ (car x) α1)
              (ψ (cdr x) α2)
              (ψ (cdr x) α3))
         (ψ x y))
```

$$(\text{implies (and (consp x)}\; (\psi\ \texttt{(car x)}\ \alpha_1)\; (\psi\ \texttt{(cdr x)}\ \alpha_2)\; (\psi\ \texttt{(cdr x)}\ \alpha_3))\; (\psi\ \texttt{x y}))$$

Instead of `(car x)` and `(cdr x)` you may use any terms whose values are smaller than `x`'s when the "test" `(consp x)` holds.

Induction Step:

$$
\begin{array}{l}
\texttt{(implies (and (consp x)} \\
\qquad\qquad\qquad\qquad\texttt{(}\psi\ \texttt{(lo x)}\ \alpha_1\texttt{)} \\
\qquad\qquad\qquad\qquad\texttt{(}\psi\ \texttt{(hi x)}\ \alpha_2\texttt{))} \\
\qquad\qquad\quad\texttt{(}\psi\ \texttt{x y))}
\end{array}
$$

`(lo` x`)` $=$ list $e \in x : e <$ `(car x)`.

`(hi` x`)` $=$ list $e \in x : e >$ `(car x)`.

Induction Principle To prove ψ, let τ be a term and let $\sigma_1, \sigma_2, \ldots$ be variable-to-term substitutions.

Base Case:

$$\texttt{(implies (not } \tau\texttt{)}\ \psi\texttt{)}$$

Induction Step:

$$\texttt{(implies (and } \tau\ \psi/\sigma_1\ \ldots\ \psi/\sigma_n\texttt{)}\ \psi\texttt{)}$$

provided

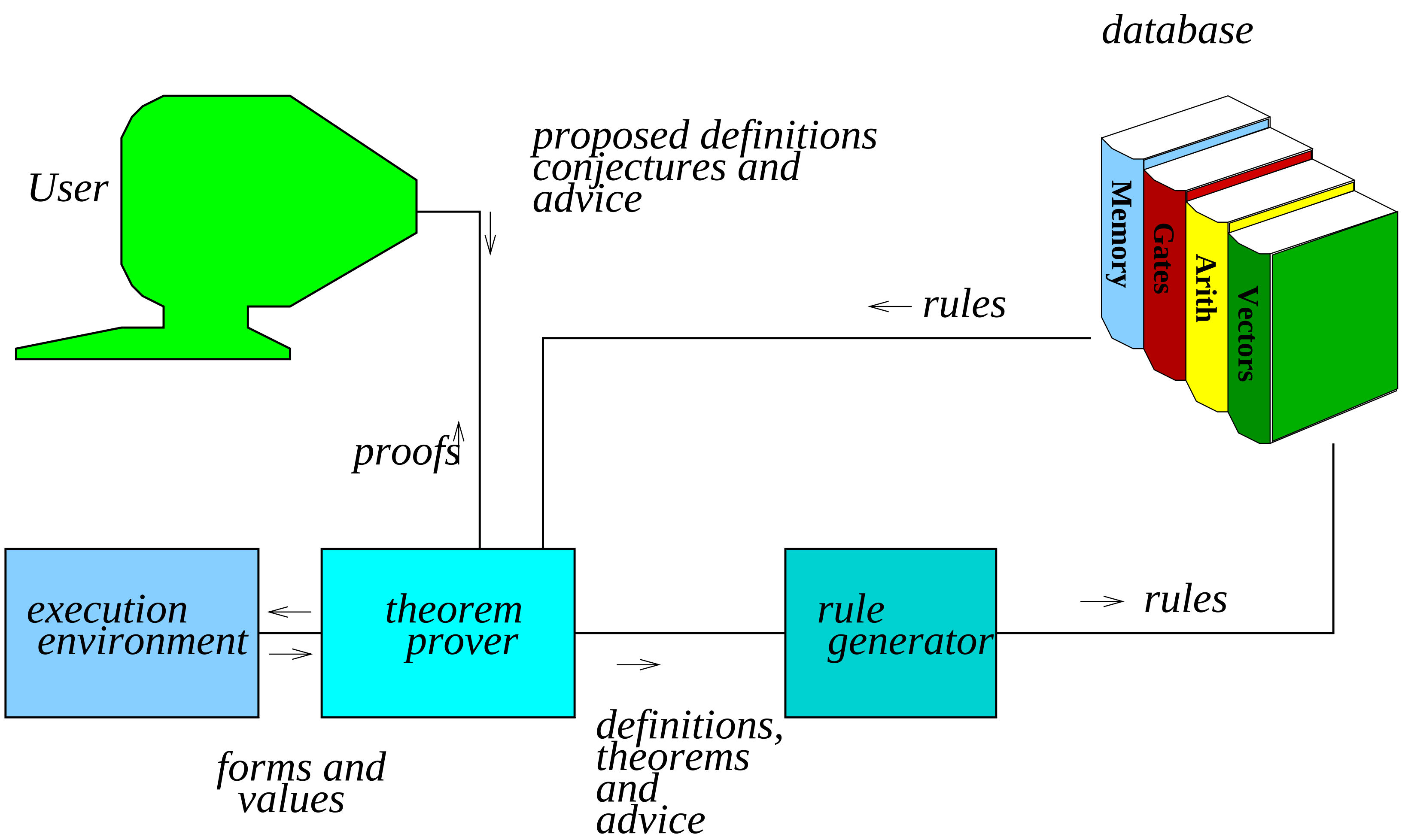
there is a term m such that

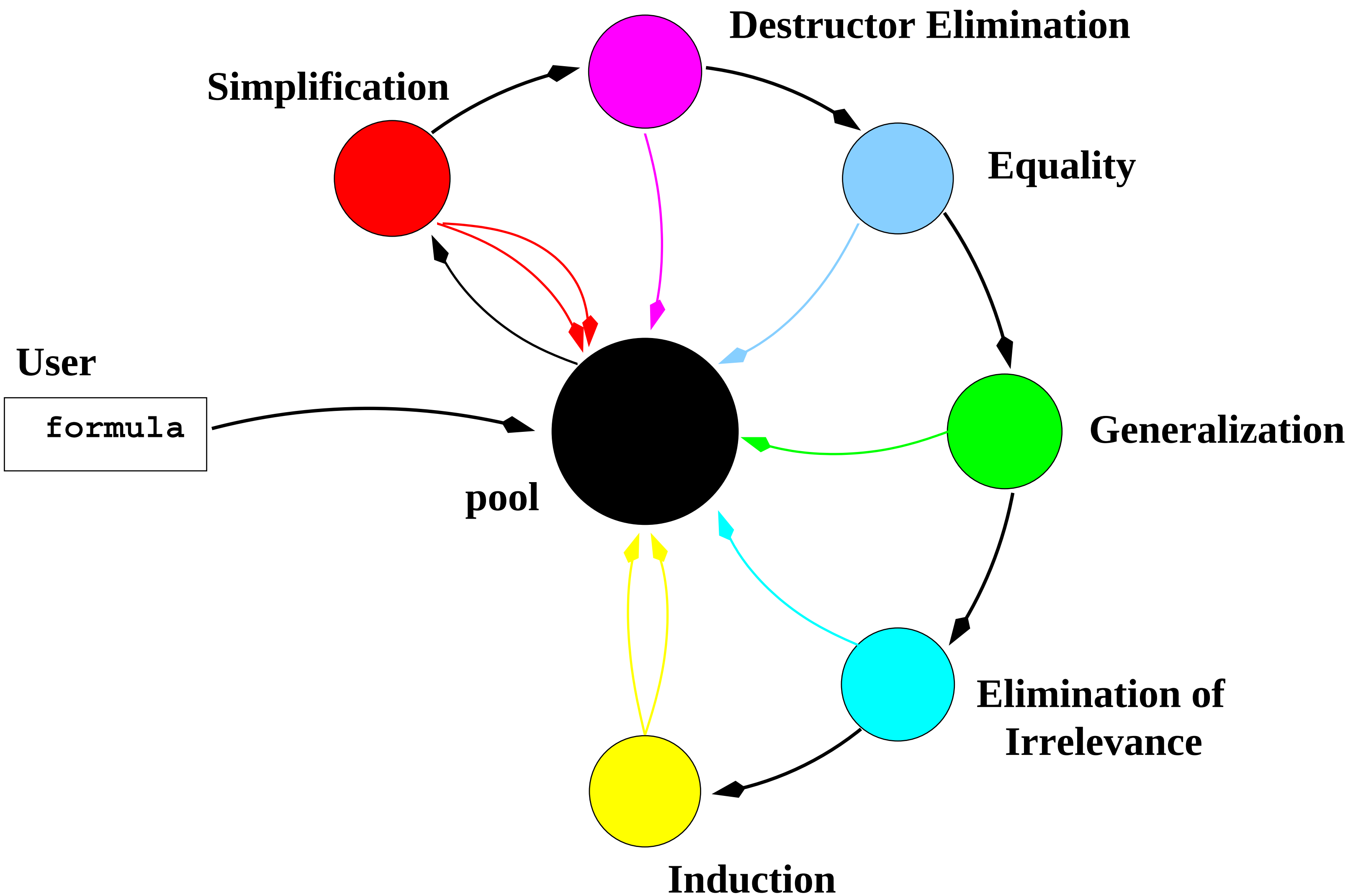
`(o-p` m`)`, and

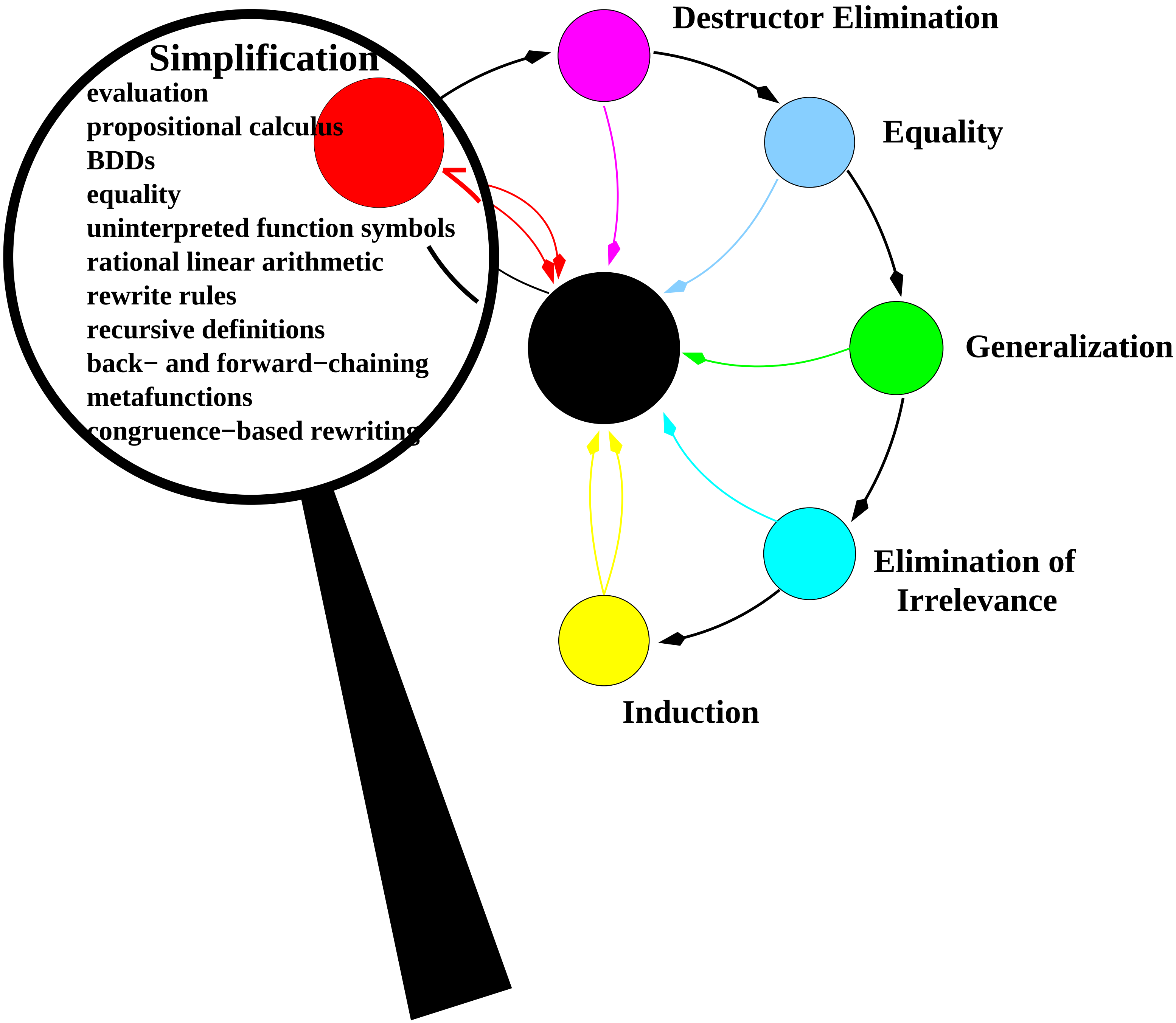
for each $1 \leq i \leq n$

```
(implies τ
         (o< m/σ_i m))
```

are theorems.


database
User
proposed definitions
conjectures and
advice
Memory
Gates
Arith
Vectors
rules
proofs
execution
environment
theorem
prover
rule
generator
rules
forms and
values
definitions,
theorems
and
advice


Destructor Elimination
Simplification
Equality
User
formula
Generalization
pool
Elimination of Irrelevance
Induction


Simplification
evaluation
propositional calculus
BDDs
equality
uninterpreted function symbols
rational linear arithmetic
rewrite rules
recursive definitions
back− and forward−chaining
metafunctions
congruence−based rewriting
Destructor Elimination
Equality
Generalization
Elimination of Irrelevance
Induction

Demo

Note

I will make available my solutions to the `twins` problem, the `subp` problem, and the `Hanoi` problem.

Next time I will begin to present larger systems.