Inductive Definitions via Inference Rules in ACL2

Alessandro Coglio Grant Jurgensen



Kestrel Institute

ACL2 Workshop 2026

Certain predicates are best defined inductively as the smallest ones satisfying given inference rules.

Example: reflexive and transitive closure R^* of a binary relation R.

$$\frac{R(x,y)}{R^*(x,y)}\ \text{BASE} \qquad \frac{}{R^*(x,x)}\ \text{REFL} \qquad \frac{R^*(x,y) \quad R^*(y,z)}{R^*(x,z)}\ \text{TRANS}$$

Certain predicates are best defined inductively as the smallest ones satisfying given inference rules.

Example: typing judgements $\Gamma \vdash e : \tau$ for a programming language.

$$\frac{\Gamma \vdash f : \tau \to \tau' \quad \Gamma \vdash a : \tau}{\Gamma \vdash f\ a : \tau'}\ \text{APP} \qquad \frac{\Gamma,\ x : \tau \vdash e : \tau'}{\Gamma \vdash \lambda x.e : \tau \to \tau'}\ \text{ABS} \qquad \ldots$$

Certain predicates are best defined inductively as the smallest ones satisfying given inference rules.

Example: small-step evaluation $e \Rrightarrow e'$ for a programming language.

$$\frac{a \Rrightarrow a'}{f\,a \Rrightarrow f\,a'}\ \text{ARG} \qquad \frac{}{(\lambda x.\, e)\ v \Rrightarrow e[v/x]}\ \text{BETA} \qquad \ldots$$

Certain predicates are best defined inductively as the smallest ones satisfying given inference rules.

Example: sequents $A_1, \ldots, A_n \vdash B$ in natural deduction style.

$$\frac{\bar{A}, B \vdash C}{\bar{A} \vdash B \to C}\ \text{I-IMP} \qquad \frac{\bar{A} \vdash B \wedge C}{\bar{A} \vdash B}\ \text{E-ANDL} \qquad \ldots$$

What does it mean for R^* to be defined inductively via these inference rules?

$$\frac{R(x,y)}{R^*(x,y)}\ \text{BASE} \qquad \frac{}{R^*(x,x)}\ \text{REFL} \qquad \frac{R^*(x,y) \quad R^*(y,z)}{R^*(x,z)}\ \text{TRANS}$$

Each inference rule is a universally quantified implication.

$$\forall x, y.\, R(x,y) \Longrightarrow R^*(x,y) \qquad \forall x.\, R^*(x,x) \qquad \forall x, y, z.\, R^*(x,y) \wedge R^*(y,z) \Longrightarrow R^*(x,z)$$

R^* satisfies the inference rules, and it is a subset of
any other relation that also satisfied the inference rules.

$$\mathcal{R}(r) \triangleq [\forall x, y.\, R(x,y) \Longrightarrow r(x,y)] \wedge [\forall x.\, r(x,x)] \wedge [\forall x, y, z.\, r(x,y) \wedge r(y,z) \Longrightarrow r(x,z)]$$

$$R^* \triangleq \iota r.\, \mathcal{R}(r) \wedge [\forall r'.\, \mathcal{R}(r') \Longrightarrow r \subseteq r']$$

But this is higher-order, while ACL2 is first-order.

What does it mean for R^* to be defined inductively via these inference rules?

$$\frac{R(x,y)}{R^*(x,y)}\ \text{BASE} \qquad \frac{}{R^*(x,x)}\ \text{REFL} \qquad \frac{R^*(x,y) \quad R^*(y,z)}{R^*(x,z)}\ \text{TRANS}$$

$R^*(a,b)$ holds exactly when there is a finite proof tree of instances of inference rules that concludes $R^*(a,b)$.

E.g.:

$$\frac{\dfrac{R(1,2)}{R^*(1,2)}\ \text{BASE} \qquad \dfrac{R(2,3)}{R^*(2,3)}\ \text{BASE}}{R^*(1,3)}\ \text{TRANS}$$

$$R^*(a,b) \triangleq \exists \pi.\, valid(\pi) \wedge [root(\pi) = \text{“}R^*(a,b)\text{”}]$$

And this is first-order.

$$\frac{R(x,y)}{R^*(x,y)}\ \text{BASE} \qquad \frac{}{R^*(x,x)}\ \text{REFL} \qquad \frac{R^*(x,y) \quad R^*(y,z)}{R^*(x,z)}\ \text{TRANS}$$

We define a data type of proof trees for the inference rules.

```
(fty::deftagsum r*-proof
  (:base (x y))
  (:refl (x))
  (:trans (x y z
           (premise1-proof r*-proof)
           (premise2-proof r*-proof))))
```

one constructor per rule

free variables in each rule

one proof tree per premise

The values of the free variables determine the instance of the inference rule.

$$\frac{R(x,y)}{R^*(x,y)}\ \text{BASE} \qquad \frac{}{R^*(x,x)}\ \text{REFL} \qquad \frac{R^*(x,y) \quad R^*(y,z)}{R^*(x,z)}\ \text{TRANS}$$

We define a validity predicate on proof trees.

true iff `proof` proves $R^*(a,b)$

```
(define r*-proof-validp ((proof r*-proofp) a b)
  (r*-proof-case
   proof
   :base (and (r proof.x proof.y)
              (equal a proof.x) (equal b proof.y))
   :refl (and (equal a proof.x) (equal b proof.x))
   :trans (and (equal a proof.x) (equal b proof.z)
               (r*-proof-validp proof.premise1-proof
                                proof.x proof.y)
               (r*-proof-validp proof.premise2-proof
                                proof.y proof.z))))
```

non-R^* premise

constraints on arguments of R^* in conclusion

one case per rule

R^* premise

$$\frac{R(x,y)}{R^*(x,y)}\ \text{BASE} \qquad \frac{}{R^*(x,x)}\ \text{REFL} \qquad \frac{R^*(x,y) \quad R^*(y,z)}{R^*(x,z)}\ \text{TRANS}$$

We define the predicate `r*` (i.e. R^*) in terms of existence of a valid proof tree.

```
(defun-sk r* (a b)
  (exists (proof)
          (and (r*-proof-validp proof a b)
               ...)))
```

additional technical conditions that do not alter the meaning (see the paper for details)

$$\frac{R(x,y)}{R^*(x,y)}\ \text{BASE} \qquad \frac{}{R^*(x,x)}\ \text{REFL} \qquad \frac{R^*(x,y) \quad R^*(y,z)}{R^*(x,z)}\ \text{TRANS}$$

The predicate `r*` satisfies all the inference rules, i.e. they are theorems.

```
(defthm r*-base
  (implies (r x y)
           (r* x y)))
```

```
(defthm r*-refl
  (r* x x))
```

```
(defthm r*-trans
  (implies (and (r* x y)
                (r* y z))
           (r* x z)))
```



systematic proof hints

$$\frac{R(x,y)}{R^*(x,y)}\ \text{BASE} \qquad \frac{}{R^*(x,x)}\ \text{REFL} \qquad \frac{R^*(x,y) \quad R^*(y,z)}{R^*(x,z)}\ \text{TRANS}$$

If an alternative predicate `r*-alt` also satisfies all the inference rules…

```
(encapsulate
  (((r*-alt * *) => *))
  (local (defun r*-alt (a b) (r* a b)))
  (defthm r*-alt-base (implies (r x y) (r*-alt x y)))
  (defthm r*-alt-refl (r*-alt x x))
  (defthm r*-alt-trans ...))
```

generic predicate

`r*` as witness

trivial to prove

… then `r*` is a (non-strict) subset of that predicate.

```
(defthm r*-alt-when-r*
  (implies (r* a b)
           (r*-alt a b)))
```

systematic proof hints

$$\frac{R(x,y)}{R^*(x,y)}\ \text{BASE} \qquad \frac{}{R^*(x,x)}\ \text{REFL} \qquad \frac{R^*(x,y) \quad R^*(y,z)}{R^*(x,z)}\ \text{TRANS}$$

Thus `r*` is provably the smallest predicate that satisfies the inference rules.

```
(defthm r*-base
  (implies (r x y)
           (r* x y)))
```

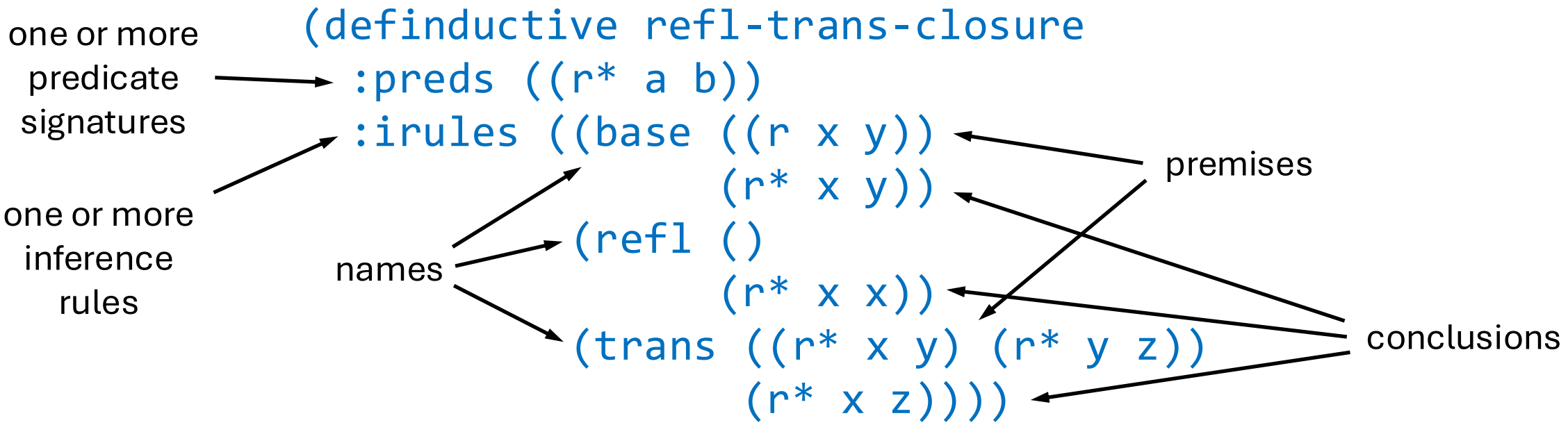
```
(defthm r*-refl
  (r* x x))
```

```
(defthm r*-trans
  (implies (and (r* x y)
                (r* y z))
           (r* x z)))
```

```
(defthm r*-alt-when-r*
  (implies (r* a b)
           (r*-alt a b)))
```

$$\frac{R(x,y)}{R^*(x,y)}\ \text{BASE} \qquad \frac{}{R^*(x,x)}\ \text{REFL} \qquad \frac{R^*(x,y) \quad R^*(y,z)}{R^*(x,z)}\ \text{TRANS}$$

The `definductive` tool generates all the previous events.



$$\frac{\Gamma \vdash f : \tau \to \tau' \quad \Gamma \vdash a : \tau}{\Gamma \vdash f\ a : \tau'}\ \text{APP} \qquad \frac{\Gamma,\ x : \tau \vdash e : \tau'}{\Gamma \vdash \lambda x.e : \tau \to \tau'}\ \text{ABS} \qquad \ldots$$

The `definductive` tool can be used to define typing judgements.

```
(definductive static-semantics
  :preds ((has-type ctxt expr type))
  :irules (...
           (app ((has-type ctxt f (fun ty ty1))
                 (has-type ctxt a ty))
                (has-type ctxt (app f a) ty1))
           (abs ((has-type (extend ctxt x ty) e ty1))
                (has-type ctxt (abs x e) (fun ty ty1)))
           ...))
```

$$\frac{a \Rrightarrow a'}{f\,a \Rrightarrow f\,a'}\ \text{ARG} \qquad \frac{}{(\lambda x.\, e)\ v \Rrightarrow e[v/x]}\ \text{BETA} \qquad \ldots$$

The `definductive` tool can be used to define small-step evaluation.

```
(definductive dynamic-semantics
  :preds ((eval expr expr1))
  :irules (...
           (arg ((eval a a1))
                (eval (app f a) (app f a1)))
           (beta ()
                 (eval (app (abs x e) v) (subst x v e)))
           ...))
```

$$\frac{\bar{A}, B \vdash C}{\bar{A} \vdash B \to C}\ \text{I-IMP} \qquad \frac{\bar{A} \vdash B \wedge C}{\bar{A} \vdash B}\ \text{E-ANDL} \qquad \ldots$$

The `definductive` tool can be used to define natural deduction with sequents.

```
(definductive sequent-calculus
  :preds ((entails formulas formula))
  :irules (...
           (i-imp ((entails (append as (list b)) c))
                  (entails as (mk-imp b c)))
           (e-andl ((entails as (mk-and b c)))
                   (entails as b))
           ...))
```

$$\frac{R(x,y)}{R^*(x,y)}\ \text{BASE} \qquad \frac{}{R^*(x,x)}\ \text{REFL} \qquad \frac{R^*(x,y) \quad R^*(y,z)}{R^*(x,z)}\ \text{TRANS}$$

The `definductive` tool provides ways to prove derived inference rules.

$$\frac{R^*(x,y) \quad R(y,z)}{R^*(x,z)}\ \text{TRANS-BASE-RIGHT}$$

$$\frac{R^*(x,y) \quad \dfrac{R(y,z)}{R^*(y,z)}\ \text{BASE}}{R^*(x,z)}\ \text{TRANS}$$

```
(defthm r*-trans-base-right
  (implies (and (r* x y)
                (r y z))
           (r* x z))
  :hints (("Goal" :in-theory (enable r*-base r*-trans))))
```

sometimes `:use` hints are needed to compose the inference rules

$$\frac{R(x,y)}{R^*(x,y)}\ \text{BASE} \qquad \frac{}{R^*(x,x)}\ \text{REFL} \qquad \frac{R^*(x,y) \quad R^*(y,z)}{R^*(x,z)}\ \text{TRANS}$$

The `definductive` tool supports both shallowly and deeply embedded proofs.

use the inference rule theorems, as in `r*-trans-base-right`

build a proof tree and prove that it is valid

shallow ⟷ deep

tradeoffs

```
(defun f (...)
  ... (r*-proof-base ...) ...)
```

proof tree constructor

```
(defthm ...
  ... (r* ...)
  :use (:instance
        r*-when-proof-validp
        (proof (f ...)))...)
```

proof tree as witness

$$\frac{R(x,y)}{R^*(x,y)}\ \text{BASE} \qquad \frac{}{R^*(x,x)}\ \text{REFL} \qquad \frac{R^*(x,y) \quad R^*(y,z)}{R^*(x,z)}\ \text{TRANS}$$

The `definductive` tool supports proofs by induction.

on proof trees ⟷ on inference rules

tradeoffs

$$\underbrace{[\forall x,y.\, R(x,y) \implies R(f(x),f(y))]}_{\text{if } f \text{ preserves } R} \implies \underbrace{[\forall x,y.\, R^*(x,y) \implies R^*(f(x),f(y))]}_{\text{then } f \text{ preserves } R^*}$$

```
(defthm r-of-f
  (implies (r x y)
           (r (f x) (f y))))
```

```
(defthm r*-of-f
  (implies (r* x y)
           (r* (f x) (f y)))
  :hints (("Goal"
           :induct (r* x y)
           ...)))
```

Both induction principles capture the idea of "all the ways in which the predicate can hold".

Theorem provers with inductive definitions via inference rules.













definductive

Theorem provers without inductive definitions via inference rules.