Lecture 3
CS 1813 – Discrete Mathematics

Truth
Inference
and
the Logical Way

Logical Inference

❑Inference (courtesy of Merriam-Webster)

- *verb* - the act of passing from one proposition, statement, or judgment considered as true to another whose truth is believed to follow from that of the former
- *noun* – a proposition arrived at by inferring

❑Formal Inference (mathematical logic)

- **Object Language** – notation for stating premises and conclusions
  - ✓ **WFFs form the object language**
- **Inference Rules** – ways to conclude new WFFs from proven WFFs
- **Metalanguage** – notation for proofs of theorems
- **Formal Inference (Proof)** – a set of assumptions together with an ordered collection of inference rules applied to reach a conclusion
- **Theorem (sequent)** – a set of assumptions and a conclusion for which there is a formal inference (that is, a proof)

Theorem

Theorem – a statement that can be proved

$$w_1, w_2, \ldots, w_n \vdash w$$

premises (WFFs) — turnstile — conclusion (WFF)

Meaning of turnstile: If the premises are true, then there is a path through the rules of inference that leads to the conclusion

Fact – a statement that is true (whether provable or not)

$$w_1, w_2, \ldots, w_n \models w$$

double turnstile

Theorems $\subset$ Facts

We will focus on <u>theorems</u> rather than mere facts

Rules of Inference

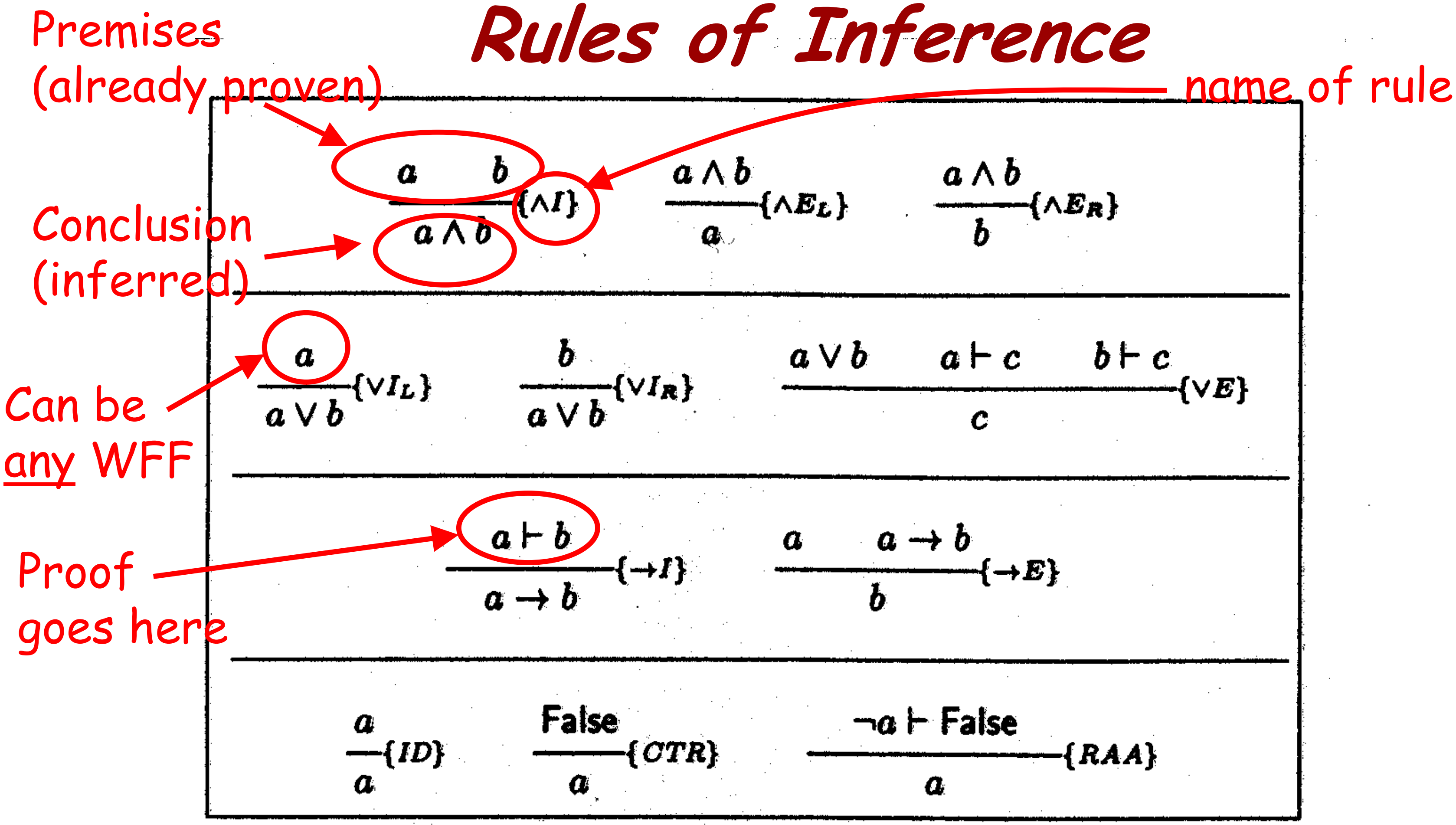
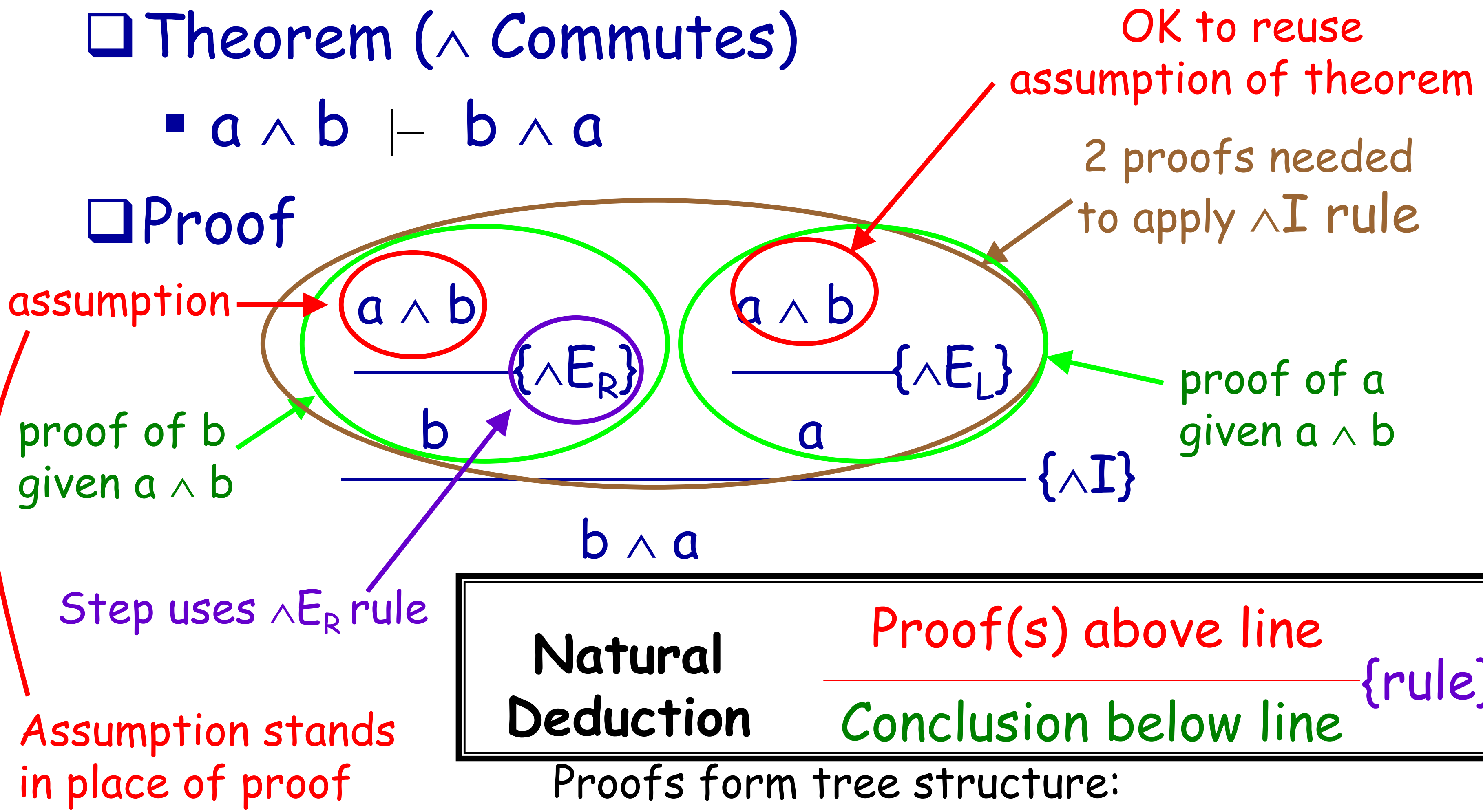


Fig 2.1, Hall/O'Donnell *Discrete Mathematics with a Computer* Springer, 2000

- Metavariables in rules stand for WFFs
- Sequents in rules stand for proofs
- Rule says: infer conclusion (bottom) if top (premises) has been proven — applying a rule is proves a theorem

Theorem and Proof

- Theorem (∧ Commutes)
  - $a \wedge b \vdash b \wedge a$
- Proof



Natural Deduction

$$\frac{\text{Proof(s) above line}}{\text{Conclusion below line}}\{\text{rule}\}$$

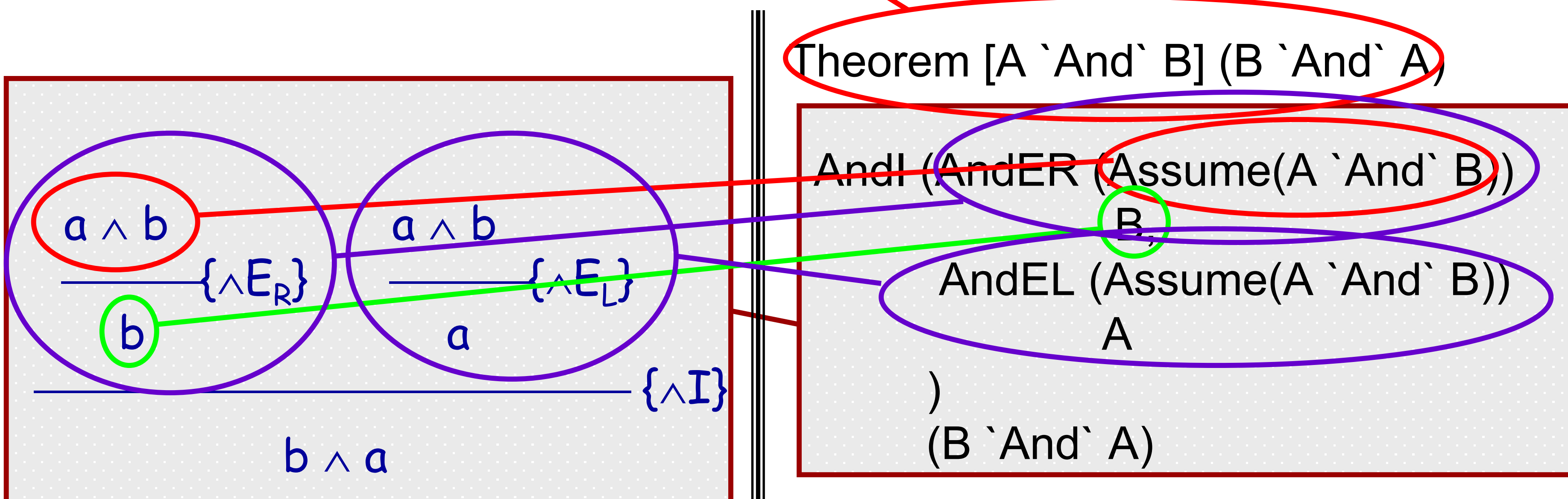
Proofs form tree structure:

- Leaves = premises (assumptions)
- Root = conclusion
- Inference rule citations = branches

$$\frac{a \qquad b}{a \wedge b}\{\wedge I\}$$

Notation for Proof Checker

- Theorem (∧ Commutes)
  - $a \wedge b \vdash b \wedge a$
- Proof



natural deduction format

Notation for Automated Proof Checker

Some Intrinsic Data Structures in Haskell

- **Sequences** (aka lists — in primitive PLs, these are "linked lists")
  - $[x_1, x_2, \ldots]$ all x's must have same type
  - Examples: [t] is the type of a sequence whose elements have type t
    - ✓ [1, 9, 3, 27] type: [Integer]
    - ✓ [And A B, Or P Q, B] type: [Prop]
- **Tuples** (like structs or records in other programming languages)
  - (c_1, c_2) pair - components may have different types
  - (c_1, c_2, c_3) 3-tuple (longer tuples OK—must be at least 2 components)
  - Examples: (t_1, t_2) means a pair where component k has type t_k
    - ✓ (7, And A B) type: (Integer, Prop)
    - ✓ (AndEL (Assume(A `And` B)) A, Assume B) type: (Proof, Proof)

      (AndEL (Assume(A `And` B)) A: Proof; Assume B: Proof)

Type Definitions in Haskell

Full definition of Prop type matches structure of WFF definition

- data Prop = A | B | And Prop Prop | Or Prop Prop …

Type names start with capital letters

Constructor names start with capital letters

These constructors require two arguments of type Prop (Prop is an inductive type)

- data Theorem = Theorem [Prop] Prop

Only one constructor for this type

First arg (premises) is a sequence whose elements have type Prop

Second arg (conclusion) is a value of type Prop

Another Type Definition

```
data Proof = Assume Prop                    |
             AndI (Proof, Proof) Prop       |
             AndEL Proof Prop               |
             AndER Proof Prop
             .
             .
             .
```

$$\frac{\text{Proof} \quad \text{Proof}}{\text{Prop}}\{\wedge I\}$$

$$\frac{\text{Proof}}{\text{Prop}}\{\wedge E_L\}$$

- **Conclusion**: each Proof constructor has a conclusion of type Prop (this represents the proven proposition)
- **Premises**: Proof constructors have one or more premises
  - Each premise is a Proof (that is, an entity of type Proof)
  - Exception: the Assume constructor has no premises
- **Constructors**: the full definition of the type Proof in the proof checker has one constructor for each inference rule

End of Lecture 3