

AFP 9 = AD together ⊕ Isomorphism (= equivalence)

(ME) ~ Exercises

"Nom, Note  
Army 16  
Rd 14"

car → [ ["Nom", "Note"];  
["Army", "16"];  
["Rd", "14"] ]

Rd → [ ["Nom", "Note"];  
["Army", "16"];  
["Rd", "14"] ]

(function) [ ["Nom", "Note2"];  
["Rd", "18"];  
["Army", "6"] ]

[ ["Nom", "Note", "Date"];  
["Army", "16", "6"];  
["Rd", "14", "18"] ]

dic / dic

obj / obj (JSON)

[ ["Nom", ["Army", ["Rd", "14"]];  
["Note", ["16", "14"]]] ]

[ ["Nom", "Army"], ["Date", "16"] ]

< / > (car item)

< \* > (public key)

exp { Set "Note" (Add # (var "Rd" (car x)))

[ ["Rd", "14"] ] < / body > < / Rnd > [ ["Rd", "14"], ["Army", "16"], ["Date", "16"] ]

[ ["Nom", ["Army", "Rd"];  
["Note", ["17", "15"]]] ]

[ ["Rd", "14"], ["Army", "16"], ["Date", "16"] ]

type EXP = car of prod  
| var of prod  
| Add of Expr Expr

type T = Expr of string  
| Sub of string (with substitution #)

let rec eval d e = ...  
type Cmd = Set of string Expr  
let rec exec d e = ...

let sub d = function  
| Expr s -> ...  
| Sub s -> ...