IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

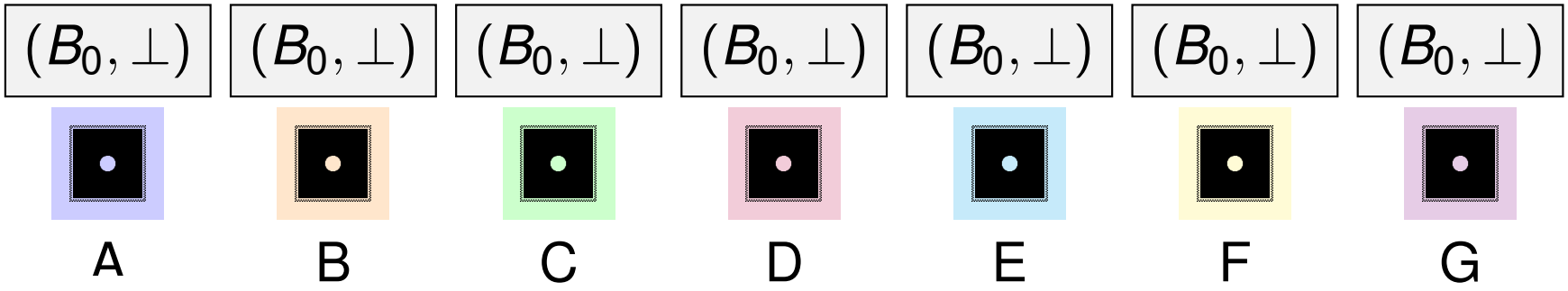
Bitcoin Blockchain as a Shared Register

E. Anceaume [1], **R. Ludinard** [2],
M. Potop-Butucaru [3], F. Tronel [4]

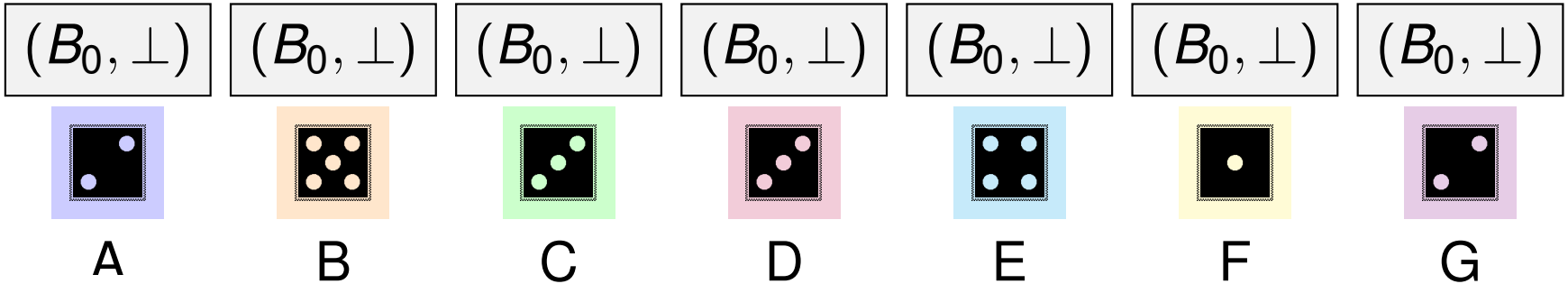
[1]CNRS / IRISA, [2]IMT Atlantique / IRISA,
[3]UMPC / LIP6, [4]CentraleSupélec / IRISA

October 16, 2017

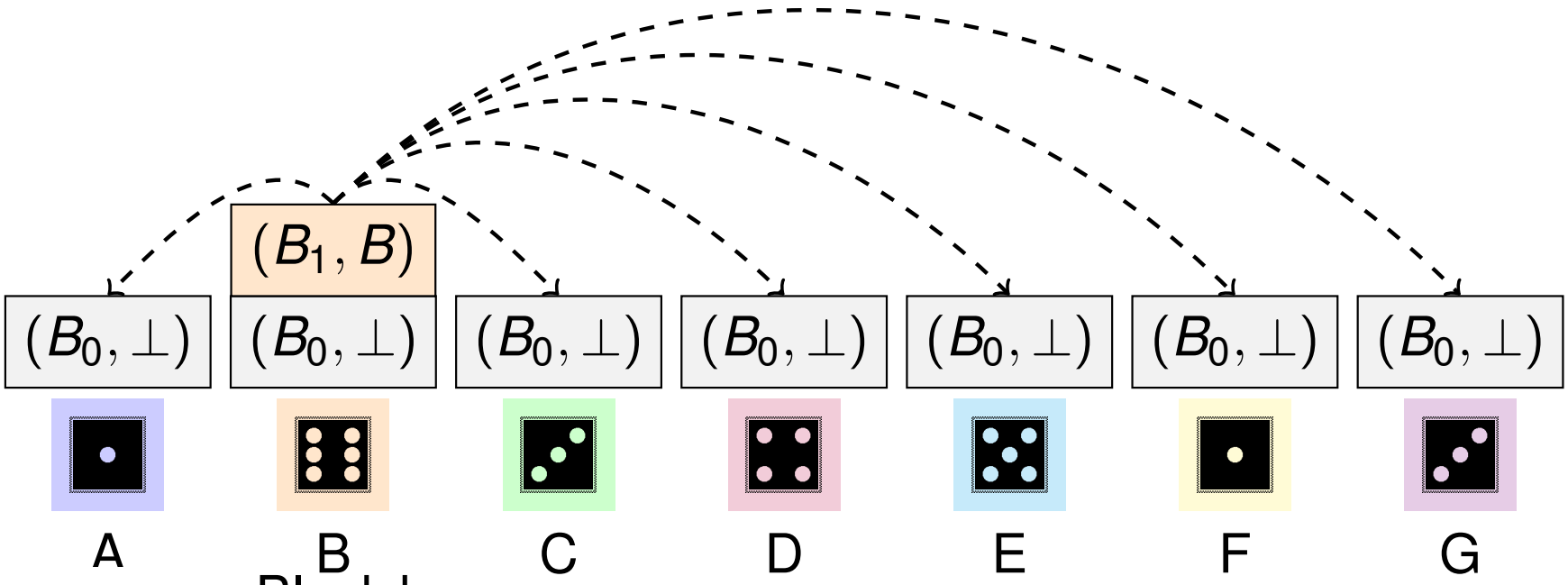
Blockchains : How it works ?



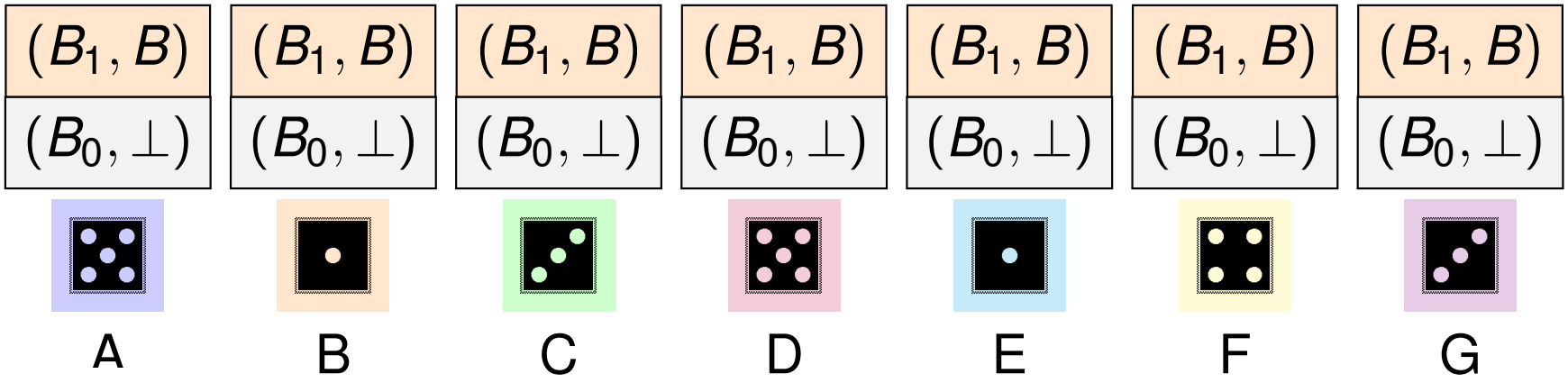
Blockchains : How it works ?



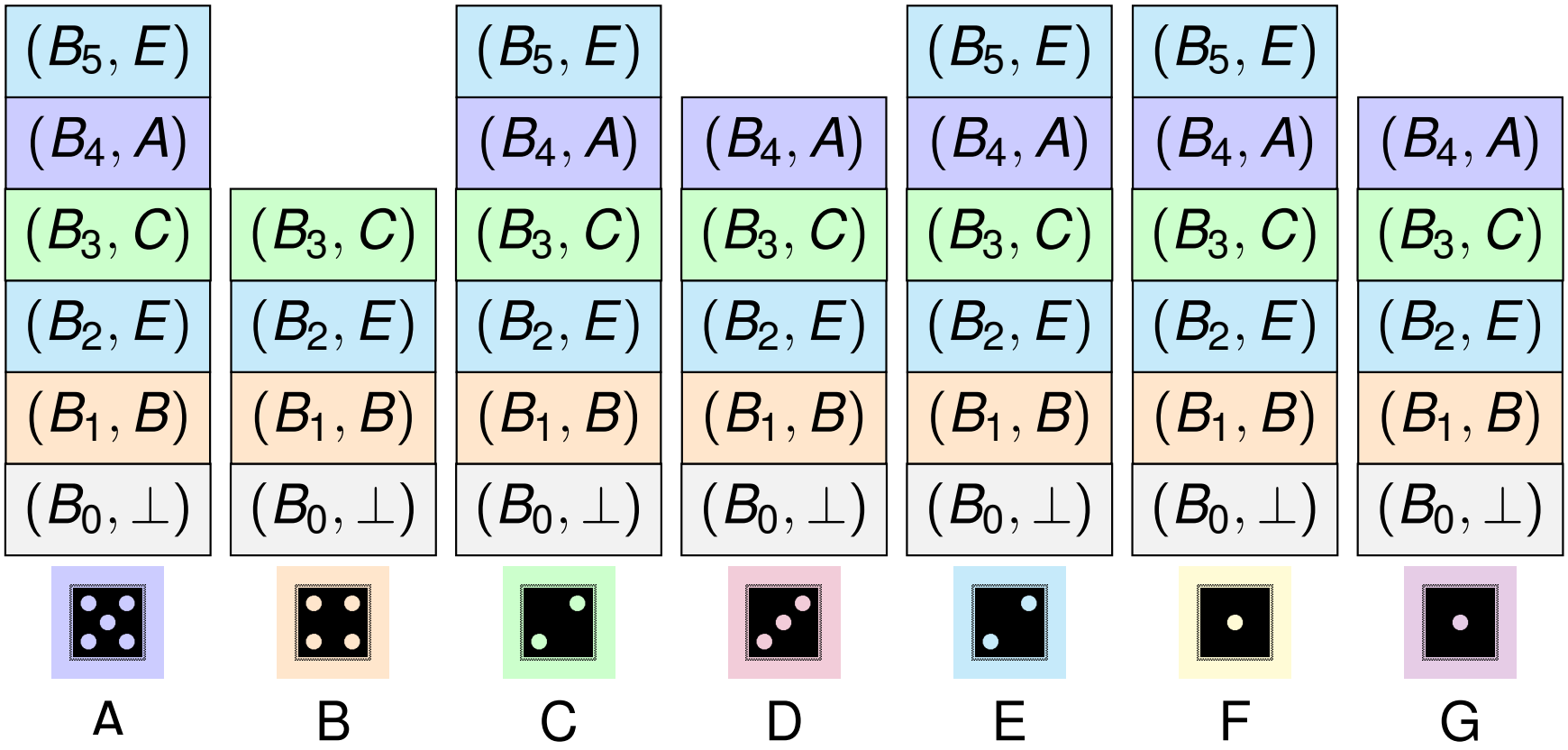
Blockchains : How it works ?



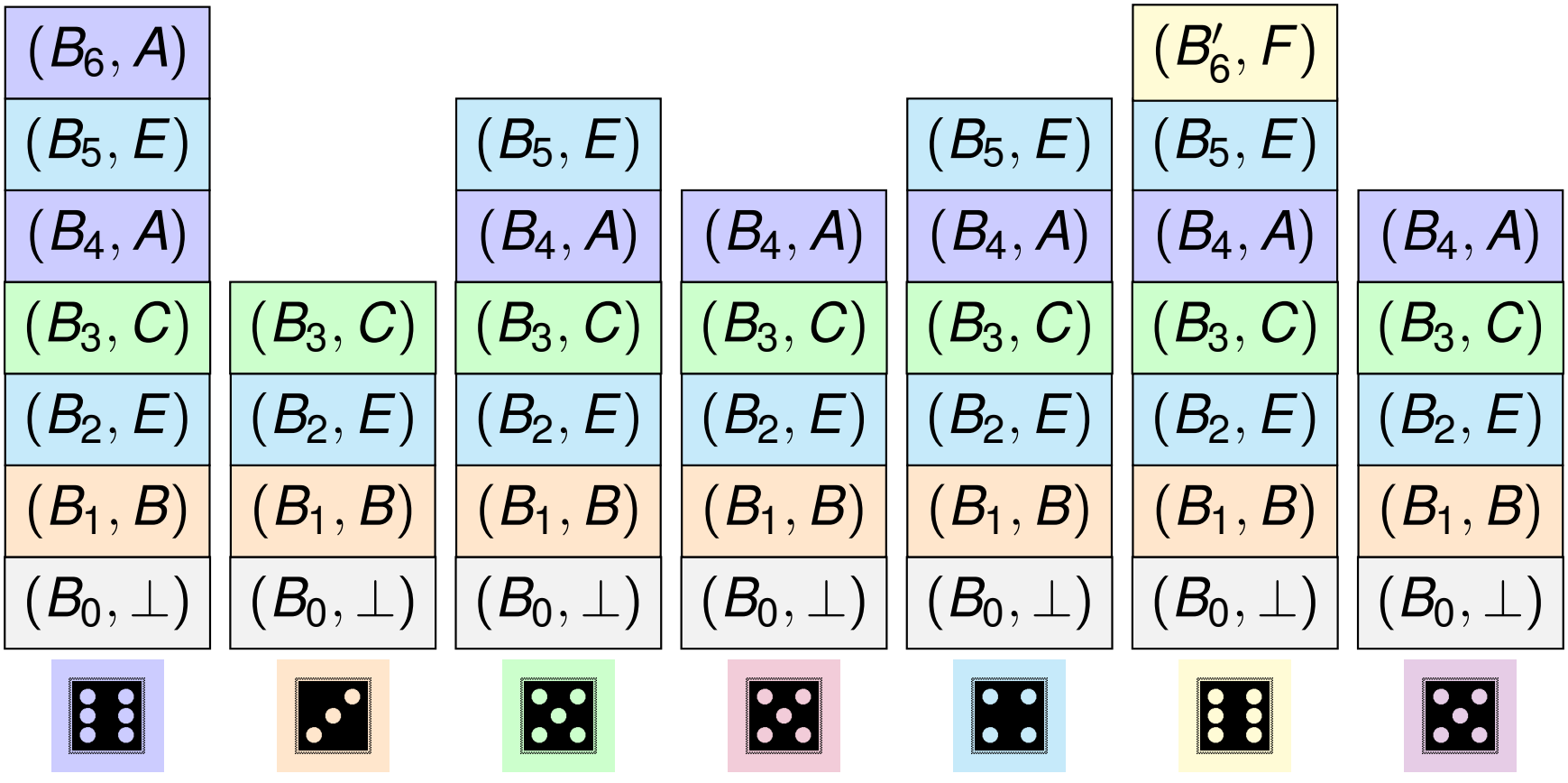
Blockchains : How it works ?



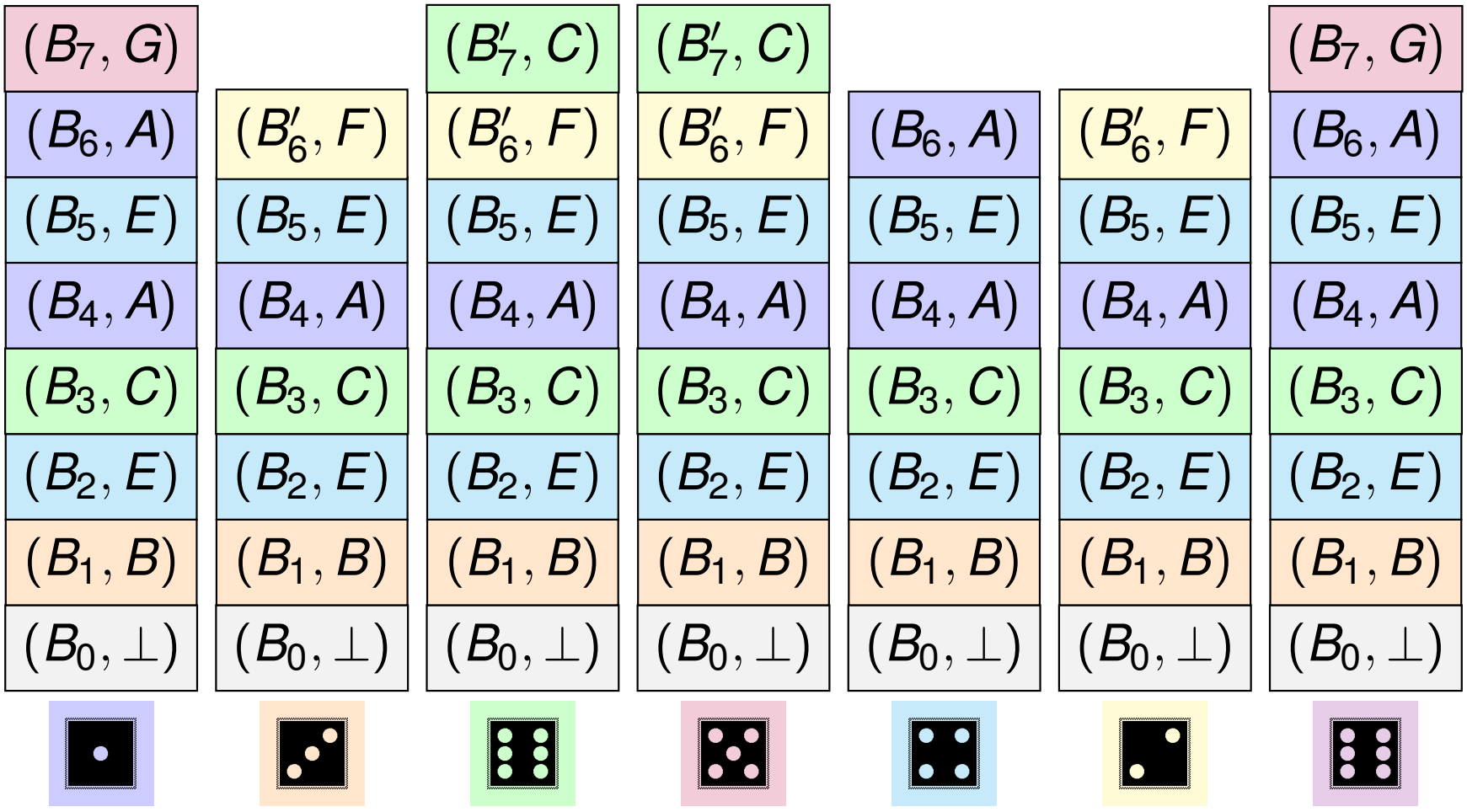
Blockchains : How it works ?



Conflict : Transient inconsistencies



Conflict : Transient inconsistencies

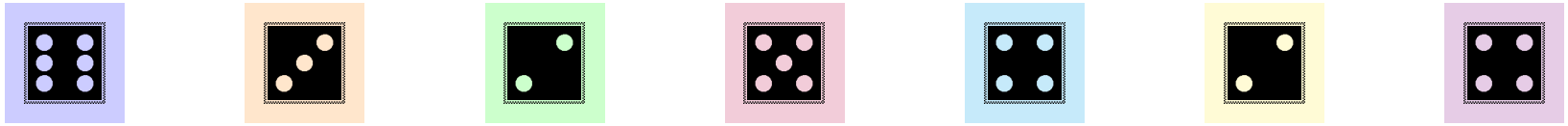


Block !

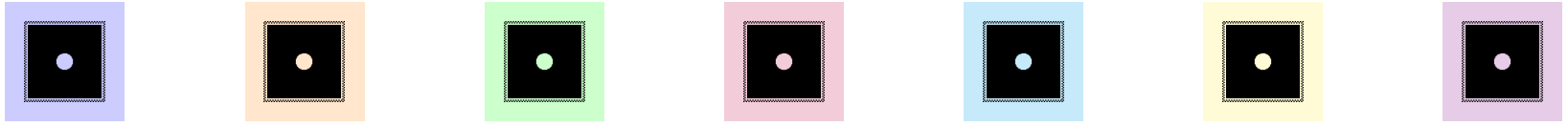
Block !

Conflict : Transient inconsistencies

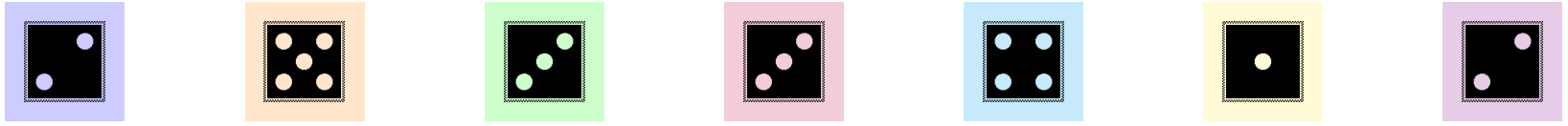
A	B	C	D	E	F	G
(B_8, A)		(B_8, A)	(B_8, A)		(B_8, A)	
(B_7, G)	(B_7, G)	(B_7, G)	(B_7, G)		(B_7, G)	(B_7, G)
(B_6, A)	(B_6, A)	(B_6, A)	(B_6, A)	(B_6, A)	(B_6, A)	(B_6, A)
(B_5, B)	(B_5, B)	(B_5, B)	(B_5, B)	(B_5, B)	(B_5, B)	(B_5, B)
(B_4, A)	(B_4, A)	(B_4, A)	(B_4, A)	(B_4, A)	(B_4, A)	(B_4, A)
(B_3, C)	(B_3, C)	(B_3, C)	(B_3, C)	(B_3, C)	(B_3, C)	(B_3, C)
(B_2, E)	(B_2, E)	(B_2, E)	(B_2, E)	(B_2, E)	(B_2, E)	(B_2, E)
(B_1, B)	(B_1, B)	(B_1, B)	(B_1, B)	(B_1, B)	(B_1, B)	(B_1, B)
$(B_0, \perp)$	$(B_0, \perp)$	$(B_0, \perp)$	$(B_0, \perp)$	$(B_0, \perp)$	$(B_0, \perp)$	$(B_0, \perp)$



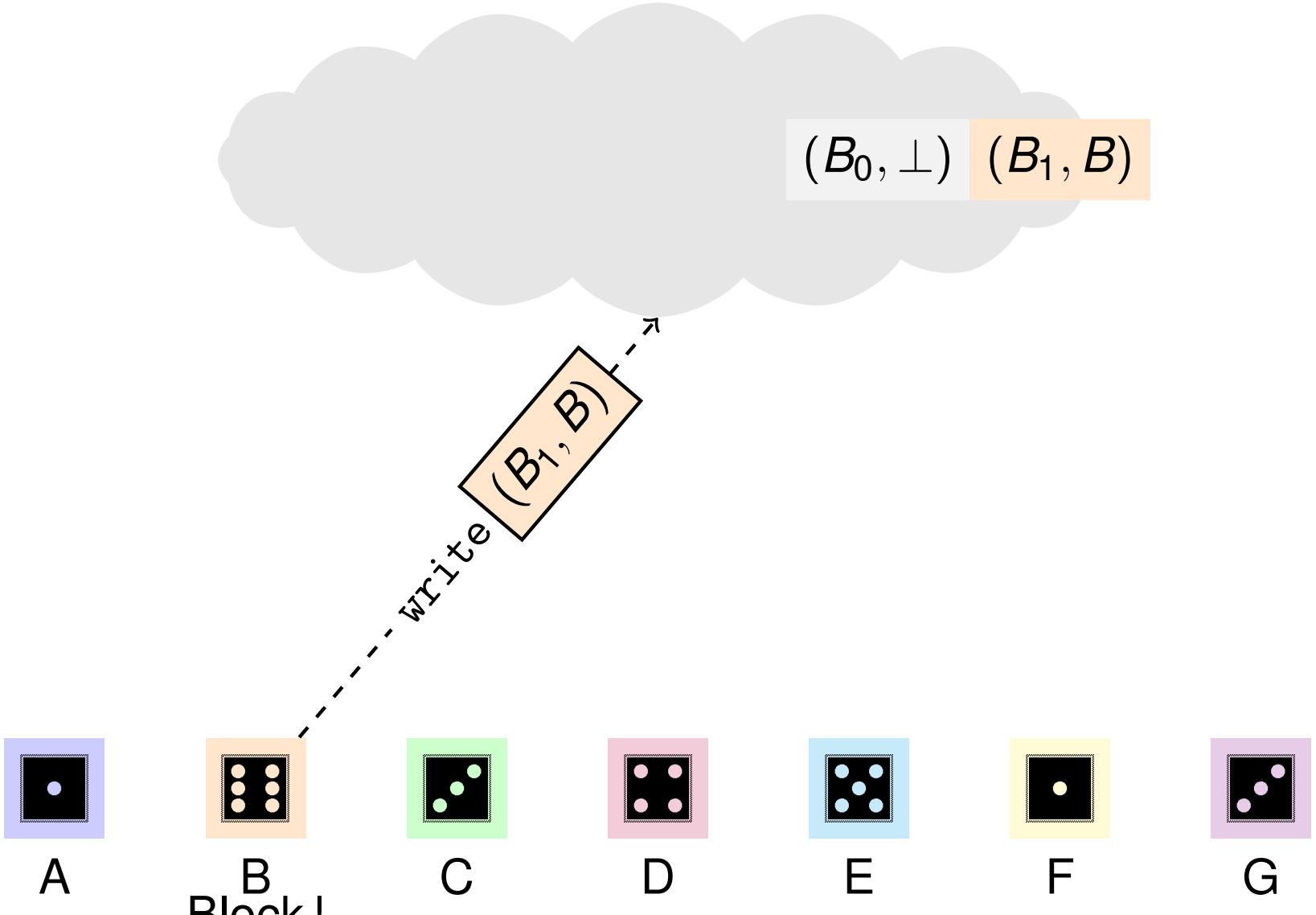
Goal : a Distributed Shared Register



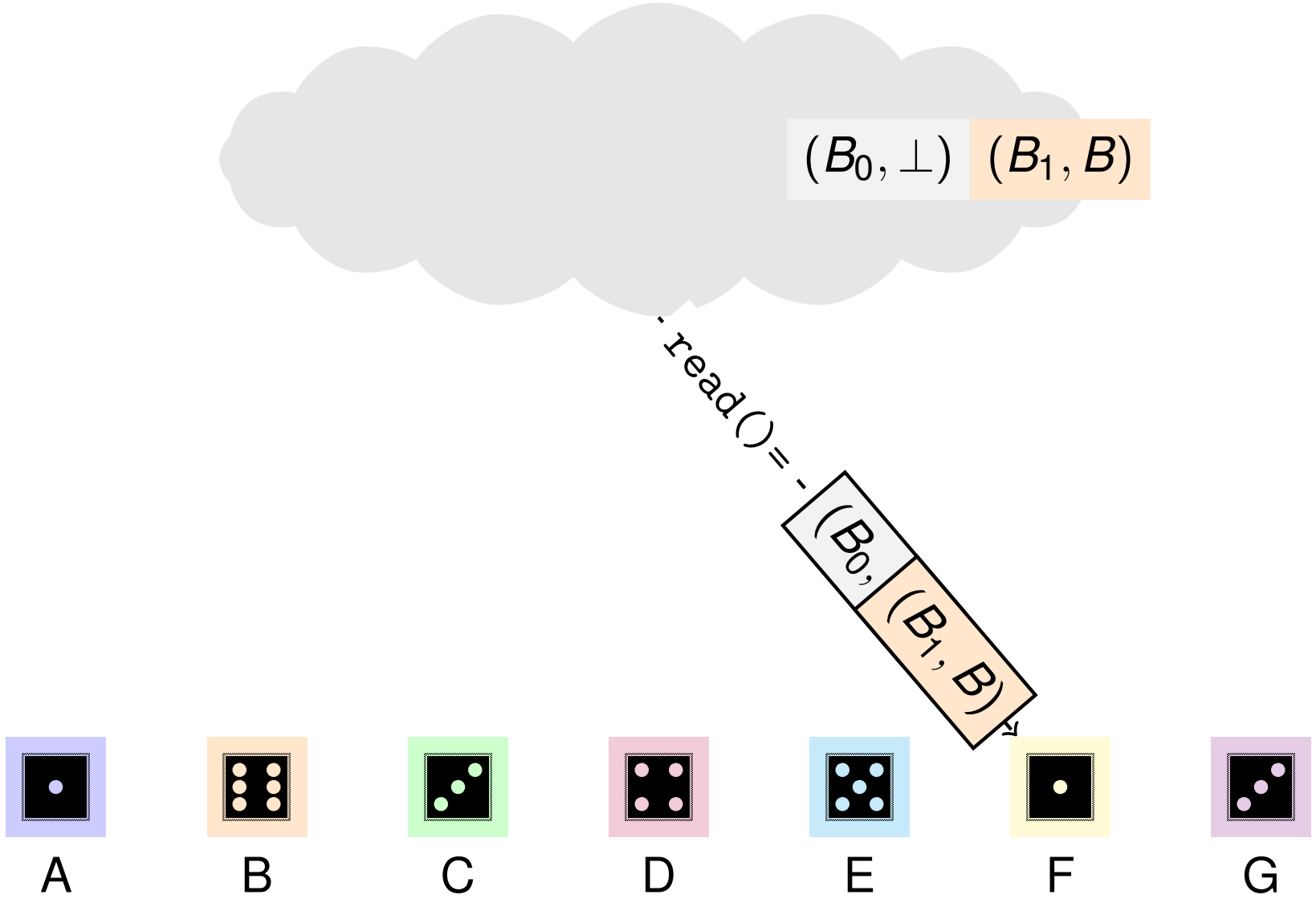
Goal : a Distributed Shared Register



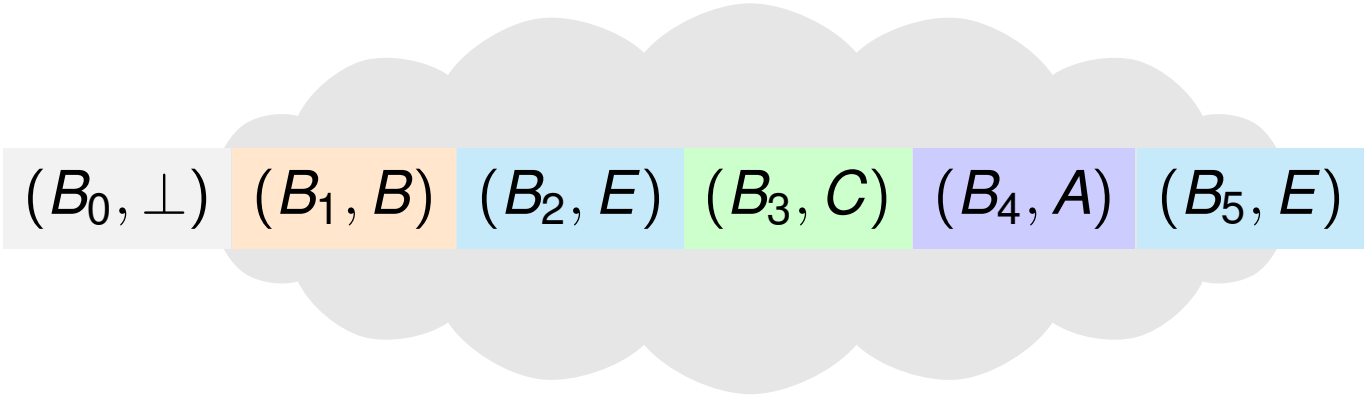
Goal : a Distributed Shared Register

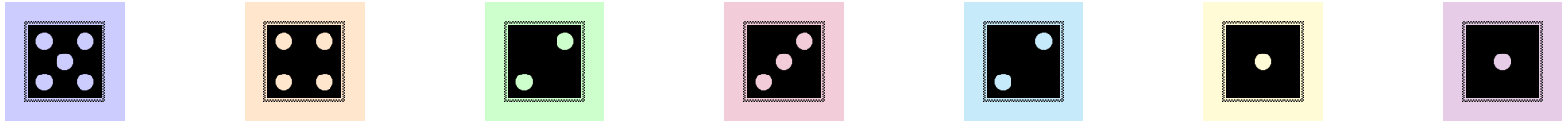


Goal : a Distributed Shared Register

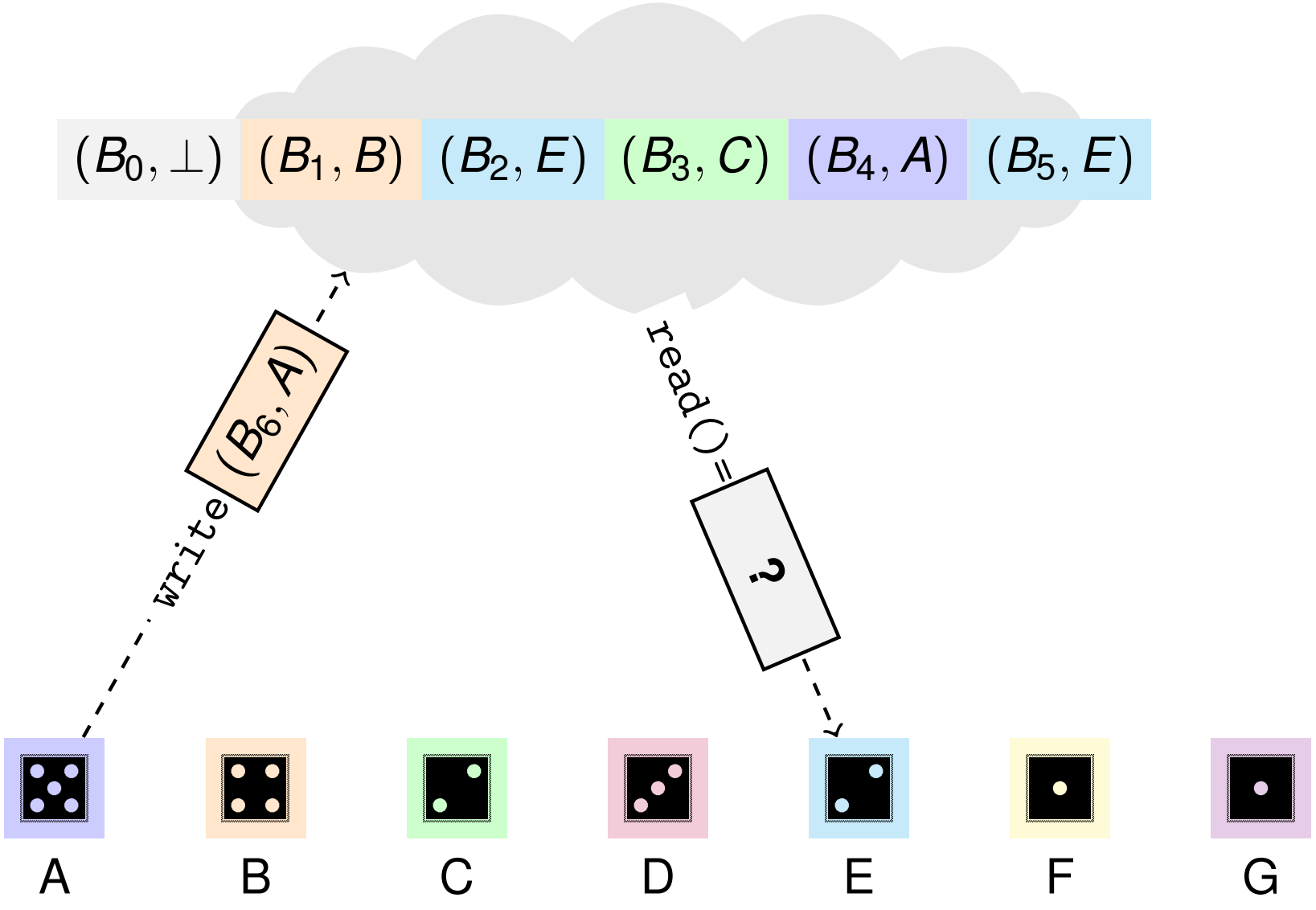


Goal : a Distributed Shared Register

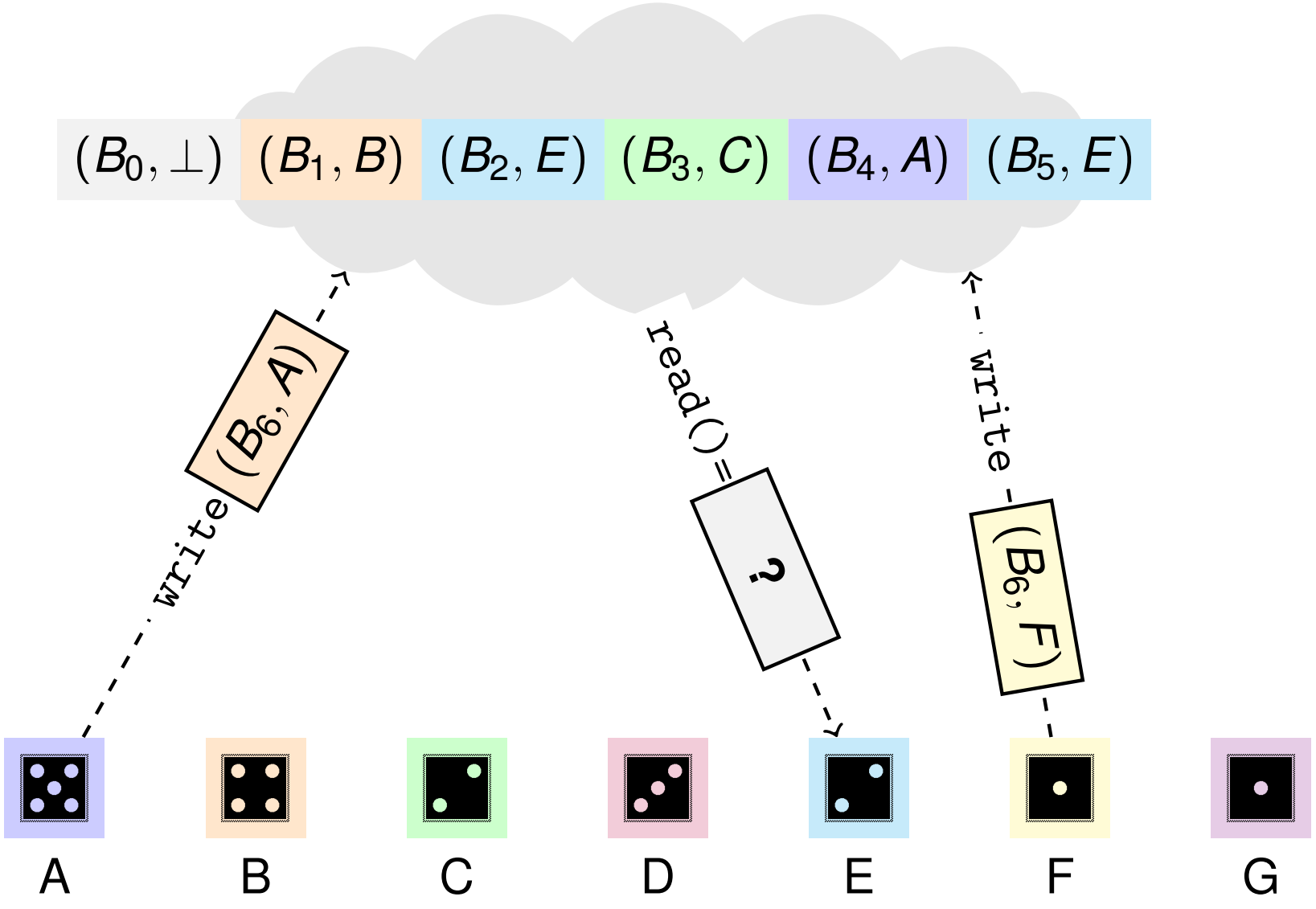




Goal : a Distributed Shared Register



Goal : a Distributed Shared Register



Liveness : Any `read` / `write` operation terminates

1. Safe register **Safety :**

$$\texttt{read}() = \begin{cases} \text{last written value,} \\ \text{any value of the register domain if concurrent } \texttt{write} \end{cases}$$

2. Regular register **Safety :**

$$\texttt{read}() = v \begin{cases} \text{last written value,} \\ \text{value written by a concurrent } \texttt{write} \end{cases}$$

3. Atomic register **Safety :**

$$(\text{Regular register safety}) + (\text{no old / new inversions})$$

Liveness : Any `read` / `write` operation terminates

1. Eventual Safe Register **Safety :** $\exists \tau > t_w$, for any `read` opertion invoked after τ, we have

$$\texttt{read}() = \begin{cases} \text{last written value,} \\ \text{any value of the register domain if concurrent } \texttt{write} \end{cases}$$

2. Eventual Regular Register **Safety :** $\exists \tau > t_w$, for any `read` opertion invoked after τ, we have

$$\texttt{read}() = v \begin{cases} \text{last written value,} \\ \text{value written by a concurrent } \texttt{write} \end{cases}$$

Distributed Ledger Registers

- `write` operations are constrained by ledger state
- Prefix of `write` history converges while no guarantees are provided on recent ones

- `write` operations are constrained by ledger state
- Prefix of `write` history converges while no guarantees are provided on recent ones

Definition

`write`($\mathcal{B}$) operation is *k-valid* at time $t_w > 0$ **if and only if** there exists an integer $k > 0$ such that for any time $t_r > t_w$ a `read` invokation at time t_r returns $\mathcal{B}'$ such that :

- $\mathcal{B}$ is a prefix of $\mathcal{B}'$
- $\texttt{length}(\mathcal{B}') \geq \texttt{length}(\mathcal{B}) + k$.

Liveness : Any `read` / `write` operation terminates

k-consistency Any `read` returns $\mathcal{B}$ whose prefix $\mathcal{B}'$ is the value of the register written by the last k-valid `write` operation.

Theorem

Such a register satisfies the **regular** *register semantic.*

Theorem

Such a register doe not satisfies the **atomic** *register semantic.*