Multithread-Chicken

Semaphore

- Ein **Semaphor** (σῆμα dt. Zeichen, φέρειν dt. tragen) ist eine Datenstruktur zur Verwaltung **beschränkter Ressourcen** in digitalen Systemen → Analogie: Semaphor in der Bahntechnik
- Die Semaphor wird als Zähler mit der maximalen Anzahl n der entsprechenden Ressource initialisiert: $s := n$
- P( ): **Prolaag** = probeer te verlagen (dt. versuche zu **senken**)
  - $s > 0$: Ressource wird beansprucht und $s := s - 1$
  - $s = 0$: Warten bis Ressource wieder verfügbar ist: $s > 0$
- V( ): **Verhogen** bzw. **vrijgeven** dt. **freigeben**: $s := s + 1$

Bei den Begriffen 'Prolaag' (probeer te verlagen), 'verhogen' bzw. 'vrijgeven' handelt es sich niederländische Worte.

Binäre Semaphore und Schlossvariablen

- Nimmt ein Semaphor ausschließlich die Werte 0 und 1 an, spricht man von einem **binären Semaphor**
- Wird ein binärer Semaphor ausschließlich von genau einem Aktivitätsträger gesenkt und freigegeben und von andren Aktivitätsträgern ausschließlich gelesen, spricht man von einem **gegenseitigen Ausschluss** (engl. **mutual exclusion** bzw. **Mutex**) oder von einer **Schlossvariablen** (**Lock**)
  - `P()` ≡ `lock()`
  - `V()` ≡ `unlock()`

Bildquelle: WPCLIPART, Gemeinfrei

Why did the multithread chicken crossed the road?

Faden 1 `S1 = ?;`

```
Chicken1() {
    printf("to ");
    [          ];
    [          ];
    printf("to ");
    [          ];
    [          ];
    printf("other ");
    [          ];
}
```

Faden 2 `S2 = ?;`

```
Chicken2() {
    [          ];
    printf("get ");
    [          ];
}
```

Faden 3 `S3 = ?;`

```
Chicken3() {
    [          ];
    printf("the ");
    [          ];
    [          ];
    printf("side "):
}
```

Why did the multithread chicken crossed the road?

Faden 1 `S1 = 0;`

```
Chicken1() {
    printf("to ");
    [          ];
    [          ];
    printf("to ");
    [          ];
    [          ];
    printf("other ");
    [          ];
}
```

Faden 2 `S2 = 0;`

```
Chicken2() {
    [          ];
    printf("get ");
    [          ];
}
```

Faden 3 `S3 = 0;`

```
Chicken3() {
    [          ];
    printf("the ");
    [          ];
    [          ];
    printf("side "):
}
```

Why did the multithread chicken crossed the road?

Faden 1 `S1 = 0;`

```
Chicken1() {
    printf("to ");
    [          ];
    [          ];
    printf("to ");
    [          ];
    [          ];
    printf("other ");
    [          ];
}
```

Faden 2 `S2 = 0;`

```
Chicken2() {
    P(&S2);
    printf("get ");
    [          ];
}
```

Faden 3 `S3 = 0;`

```
Chicken3() {
    P(&S3);
    printf("the ");
    [          ];
    [          ];
    printf("side "):
}
```

Why did the multithread chicken crossed the road?

Faden 1 `S1 = 0;`

```
Chicken1() {
    printf("to ");
    V(&S2);
    P(&S1);
    printf("to ");
    [          ];
    [          ];
    printf("other ");
    [          ];
}
```

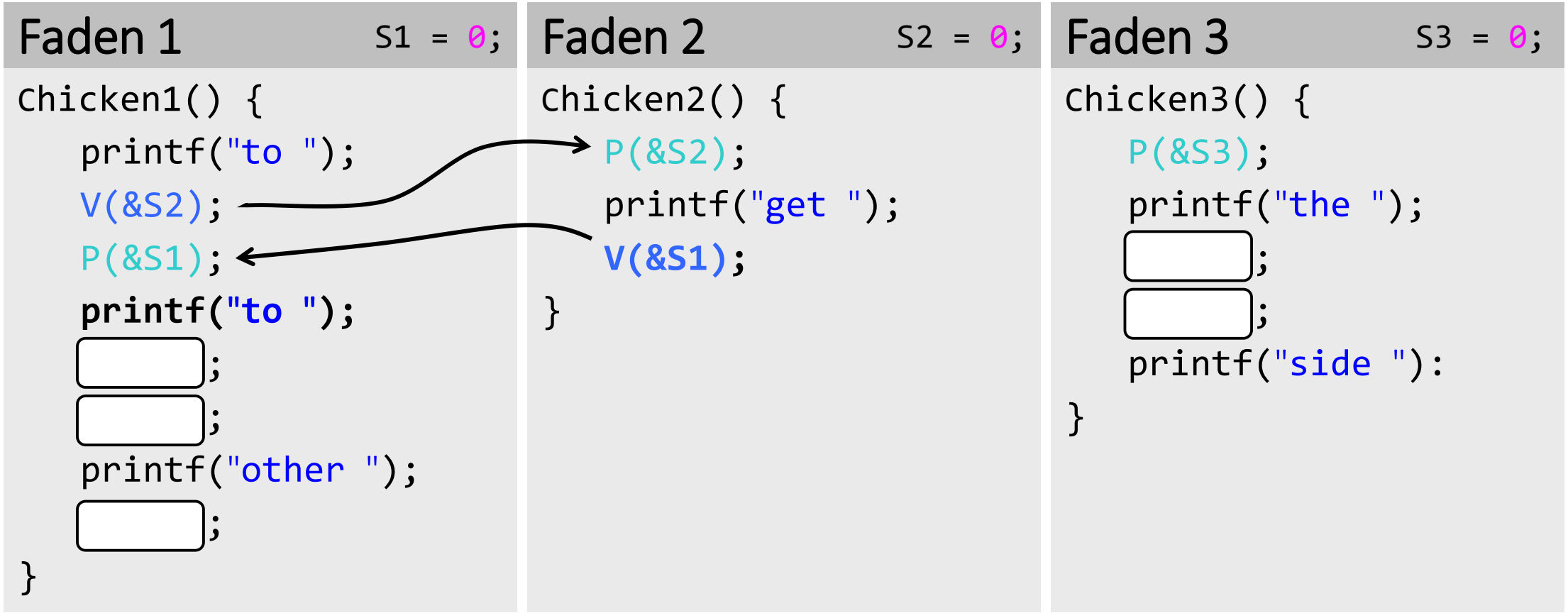
Faden 2 `S2 = 0;`

```
Chicken2() {
    P(&S2);
    printf("get ");
    [          ];
}
```

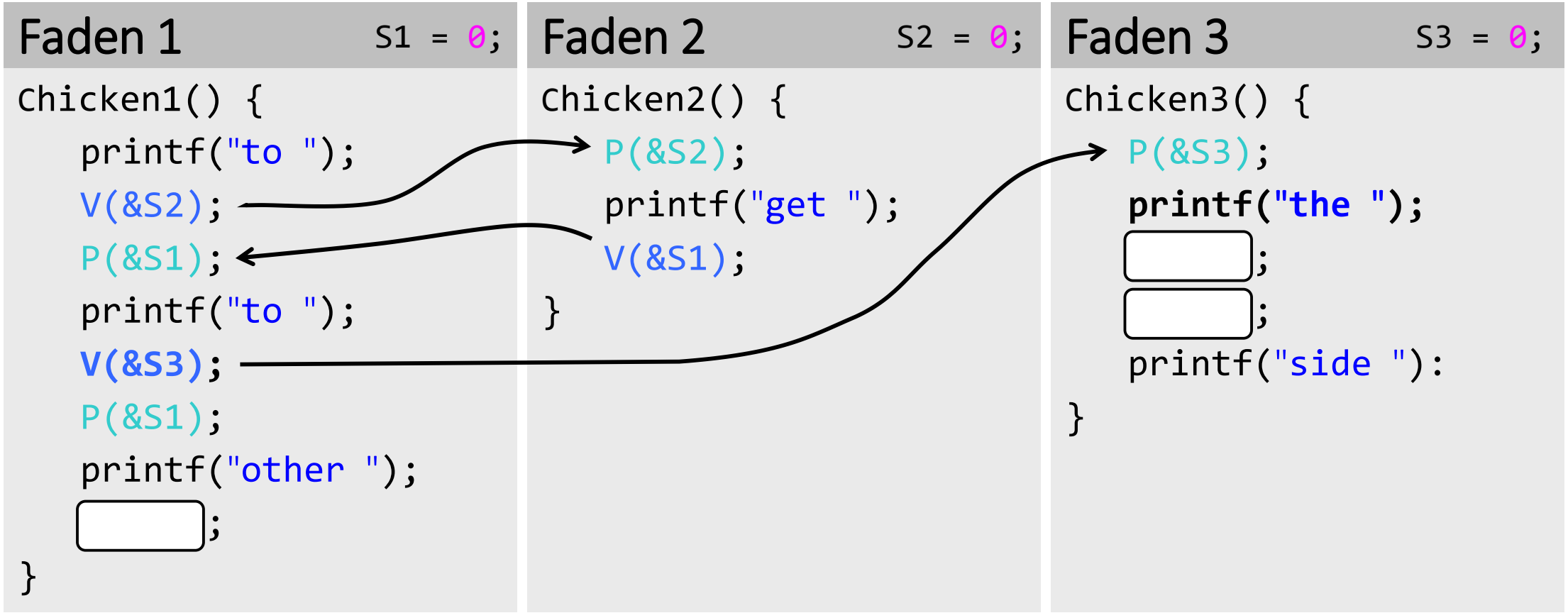
Faden 3 `S3 = 0;`

```
Chicken3() {
    P(&S3);
    printf("the ");
    [          ];
    [          ];
    printf("side "):
}
```

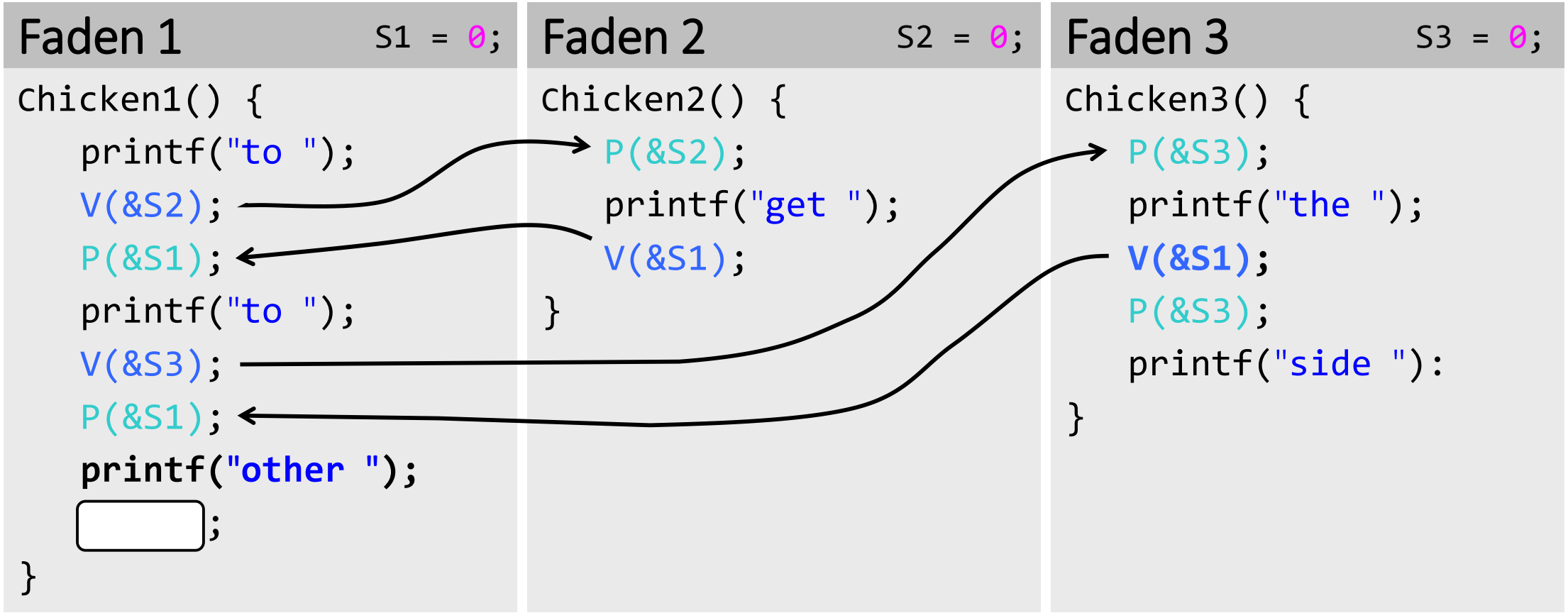
Why did the multithread chicken crossed the road?



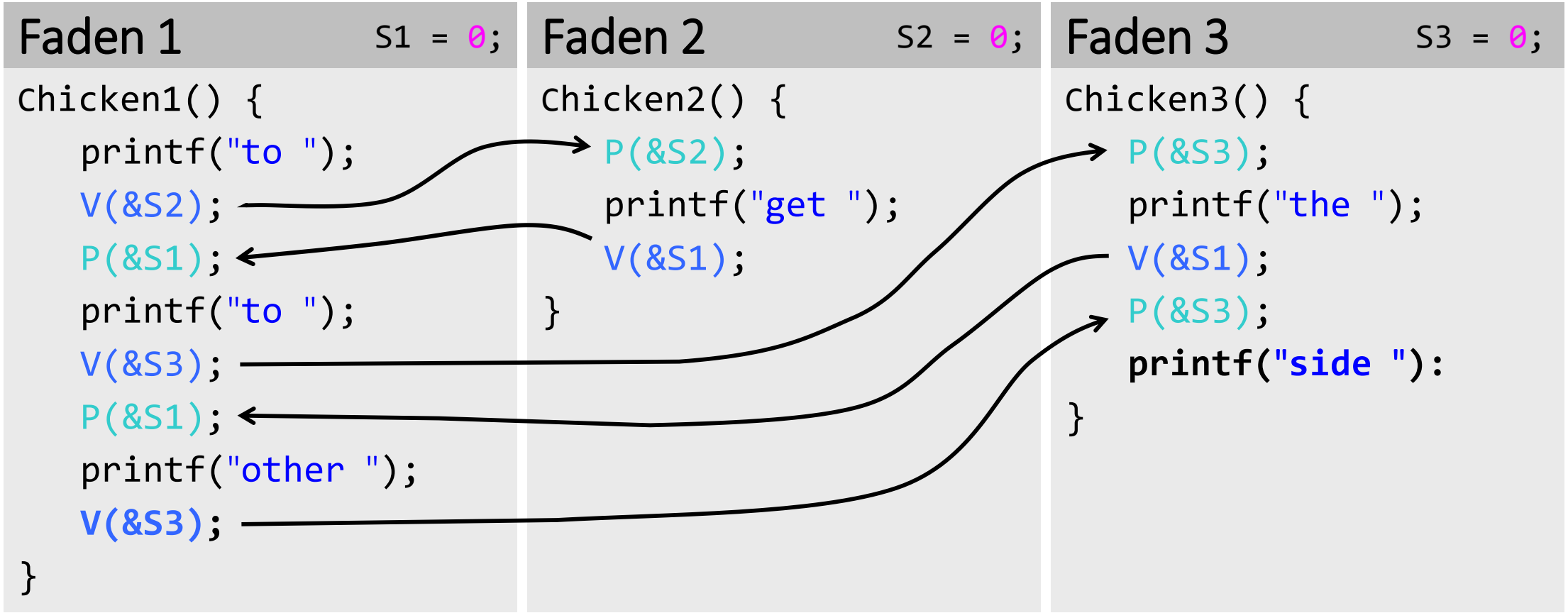
Why did the multithread chicken crossed the road?



Why did the multithread chicken crossed the road?



Why did the multithread chicken crossed the road?



Why did the multithread chicken crossed the road?