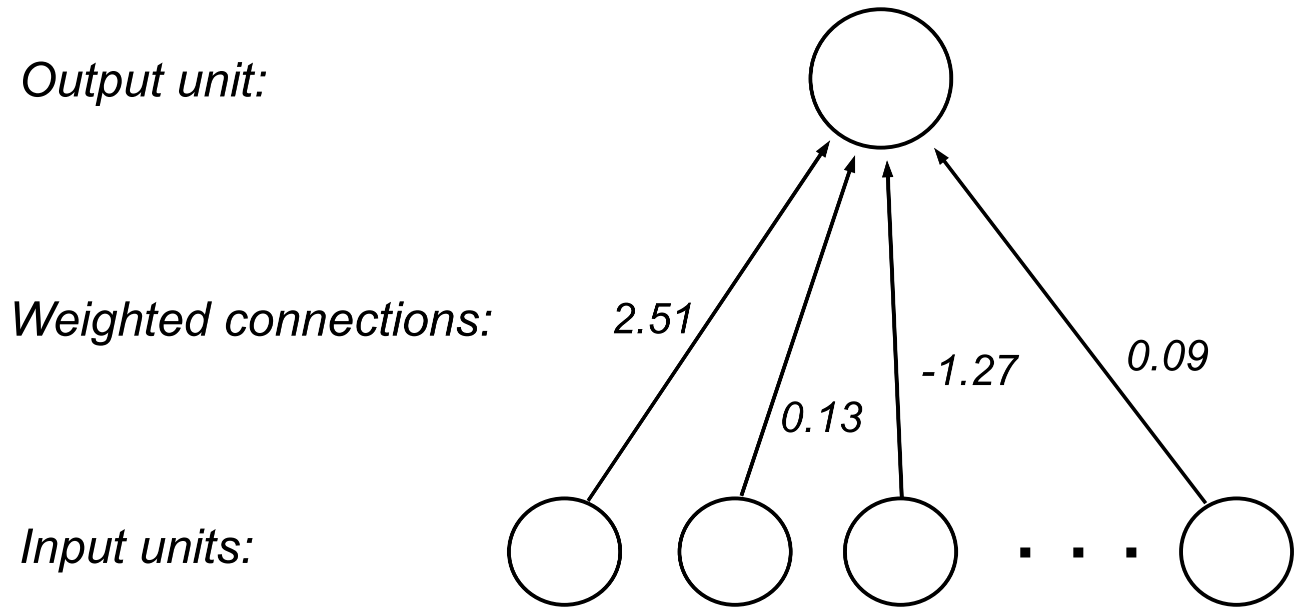
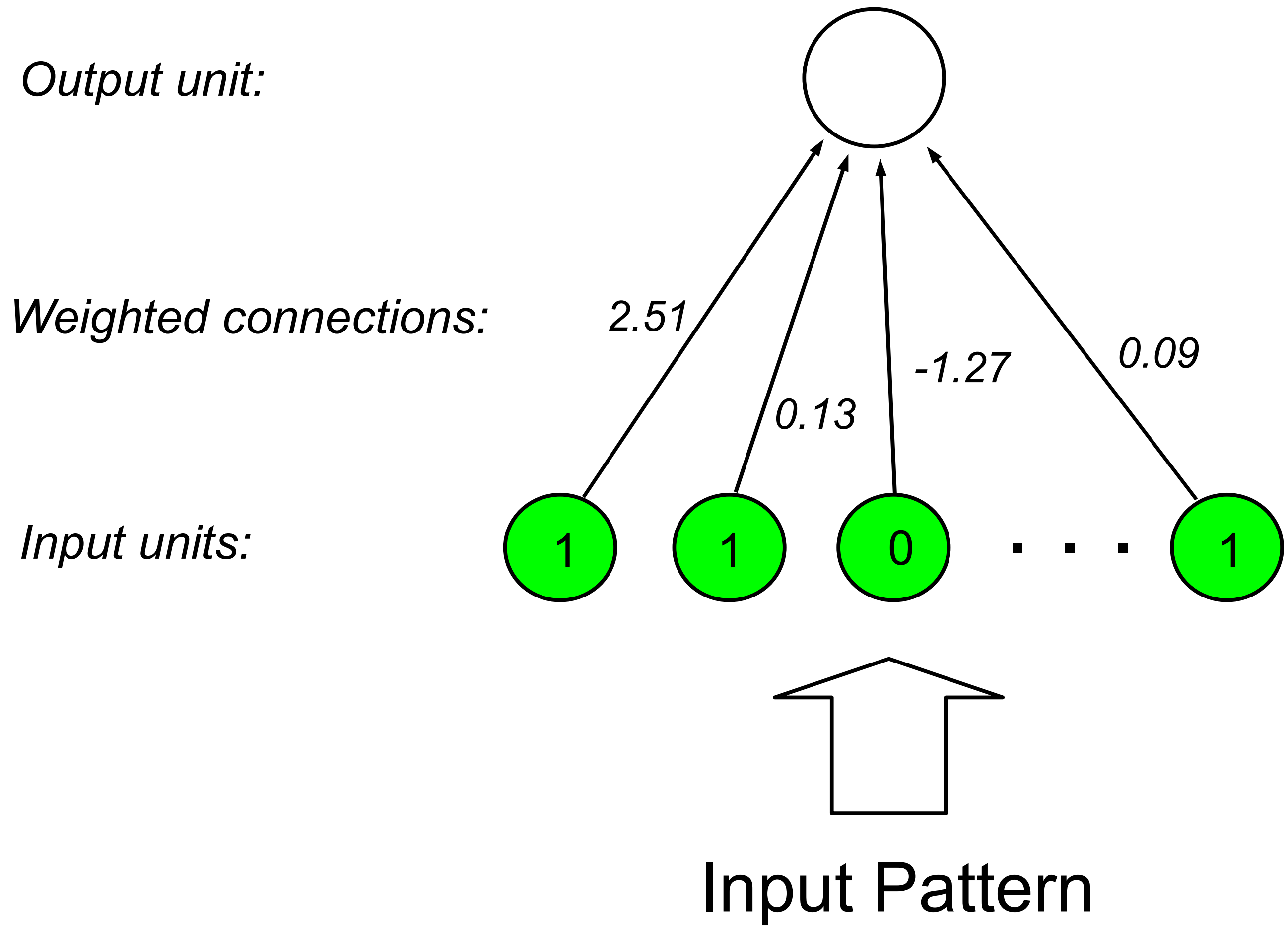
# Artificial Neurons: A Recap

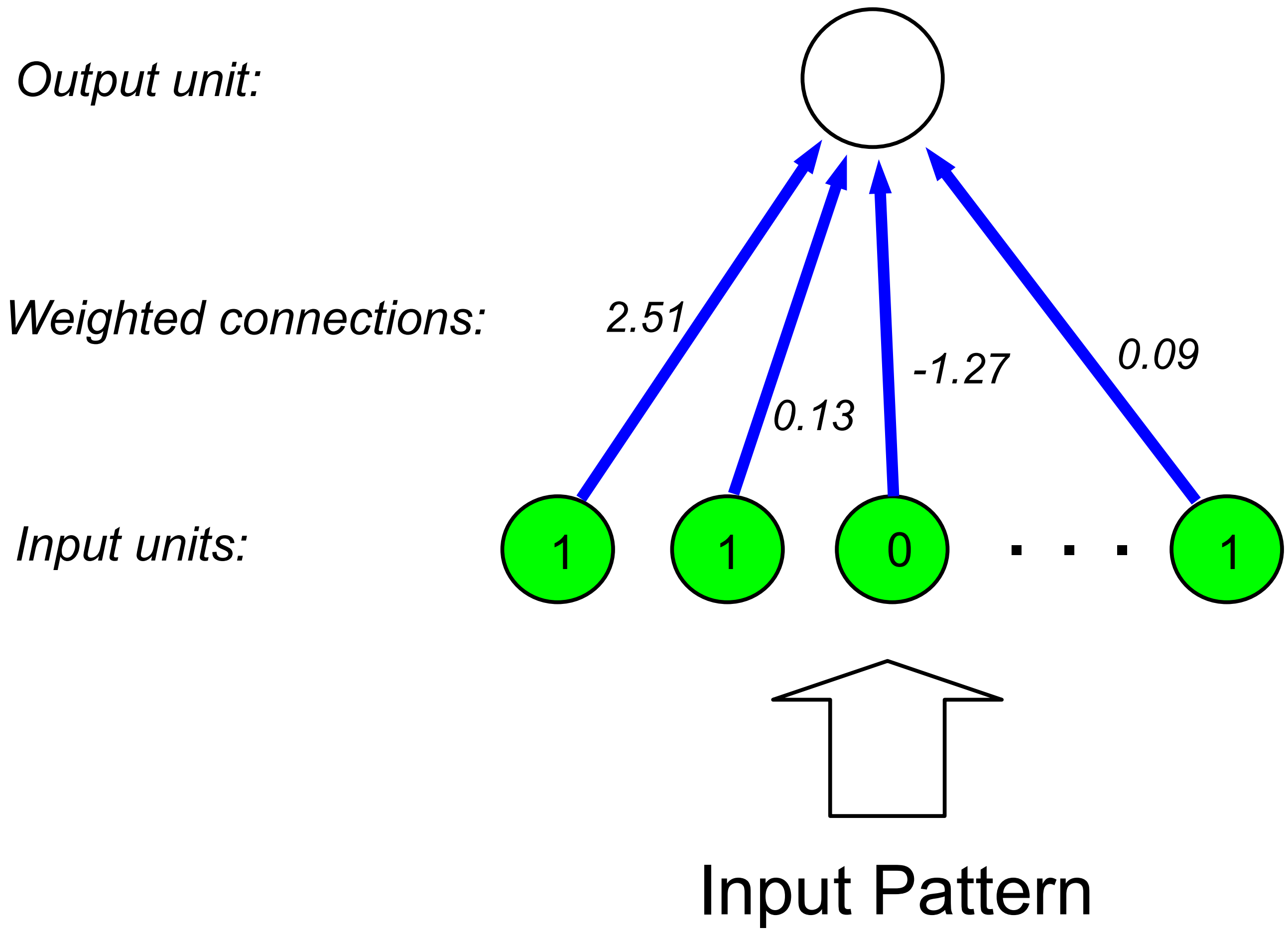


# Artificial Neurons: A Recap



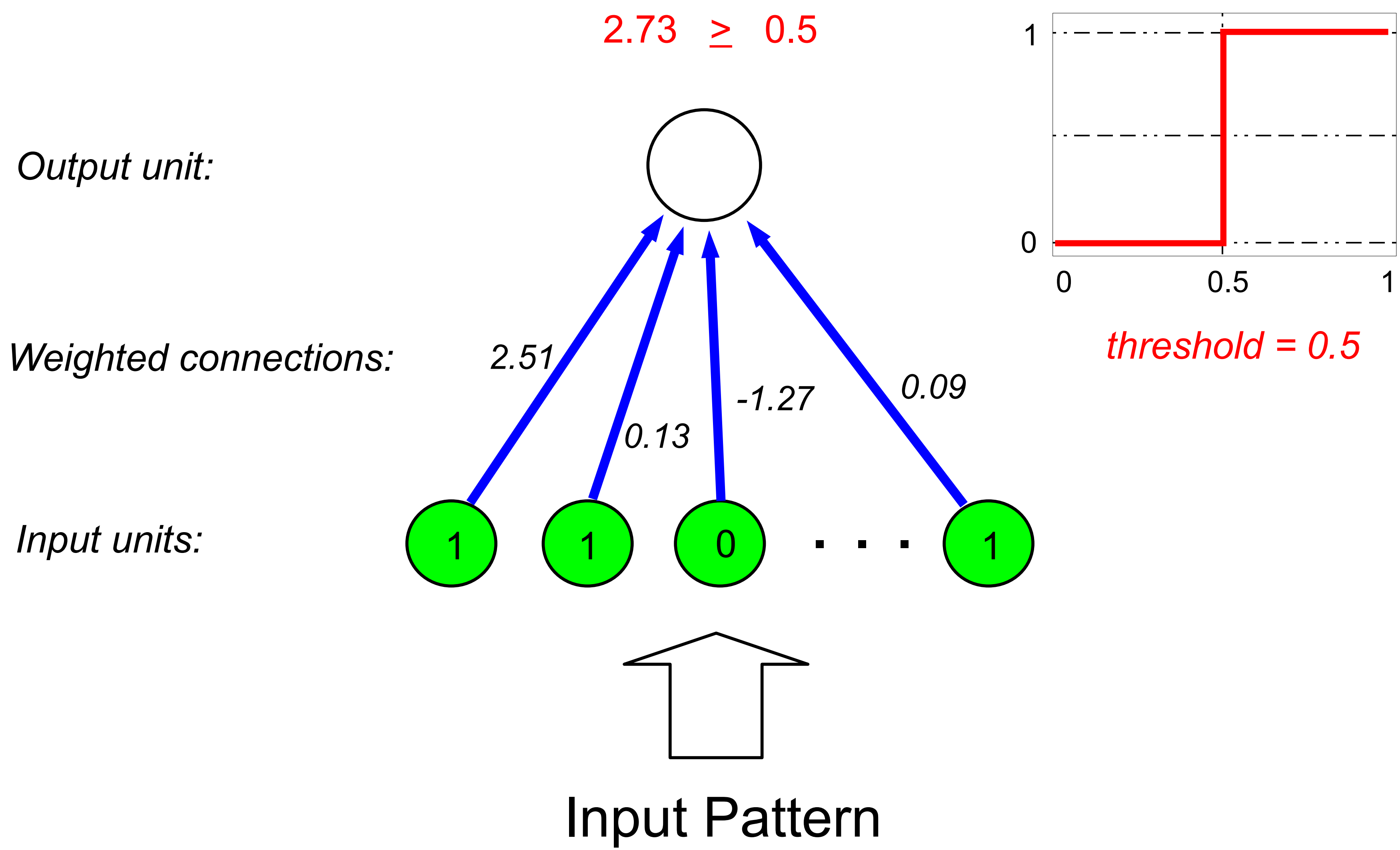
# Artificial Neurons: A Recap

$$1 \times 2.51 + 1 \times 0.13 + 0 \times -1.27 + 1 \times 0.09 = 2.73$$

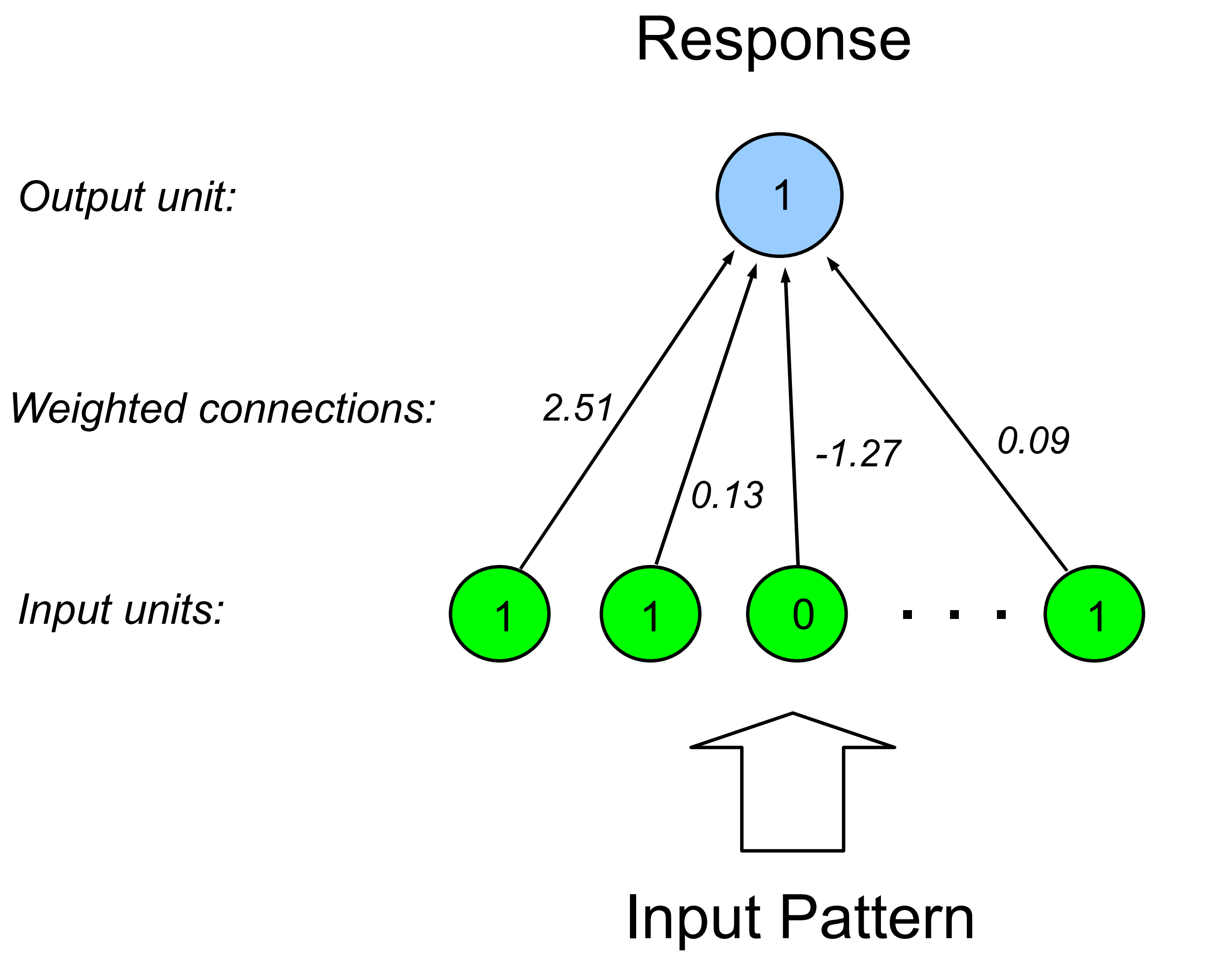


# Artificial Neurons: A Recap

$$1 \times 2.51 + 1 \times 0.13 + 0 \times -1.27 + 1 \times 0.09 = 2.73$$

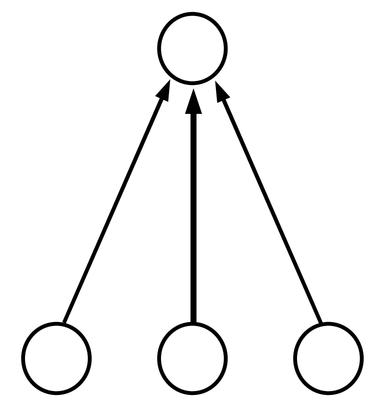


# Artificial Neurons: A Recap



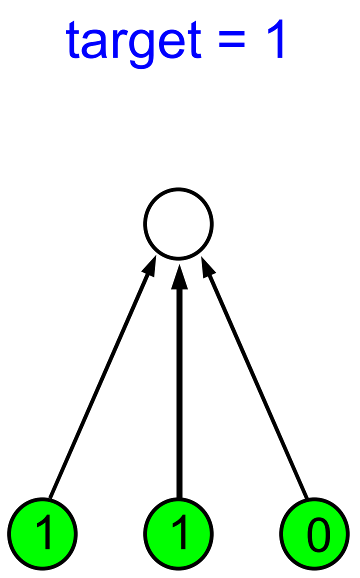
# Perceptrons

- Binary threshold neurons
- Studied by Frank Rosenblatt of Cornell in early 1960's
- Perceptron training procedure



# Perceptrons

- Binary threshold neurons
- Studied by Frank Rosenblatt of Cornell in early 1960's
- Perceptron training procedure
    1. present an input pattern

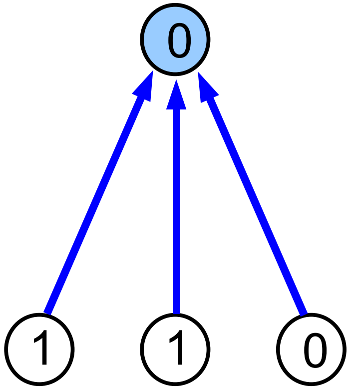


# Perceptrons

- Binary threshold neurons
- Studied by Frank Rosenblatt of Cornell in early 1960's
- Perceptron training procedure
  1. present an input pattern
  2. compute output value

     *output* = $\Theta$(sum of: *inputs* $\times$ *weights*)

target = 1

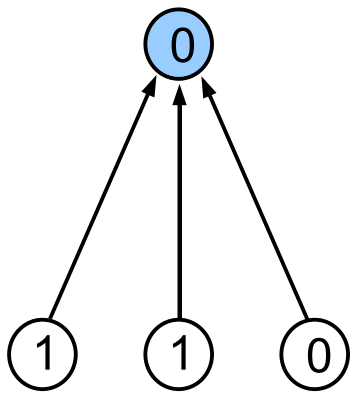


# Perceptrons

- Binary threshold neurons
- Studied by Frank Rosenblatt of Cornell in early 1960's
- Perceptron training procedure
  1. present an input pattern
  2. compute output value

     *output* = $\Theta$(sum of: *inputs* $\times$ *weights*)
  3. compare output to target value

     *error = target - output*



# Perceptrons

- Binary threshold neurons
- Studied by Frank Rosenblatt of Cornell in early 1960's
- Perceptron training procedure

  1. present an input pattern
  2. compute output value

     *output* = $\Theta$(sum of: *inputs* $\times$ *weights*)
  3. compare output to target value

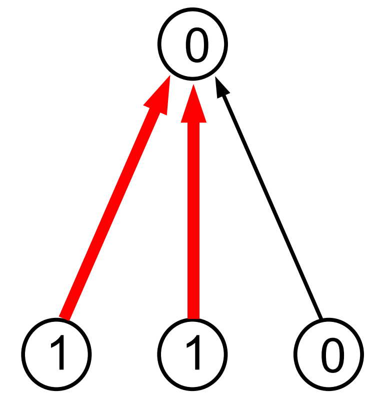
     *error* = *target* - *output*
  4. if incorrect, adjust weights

     *weight_adjust* = $\varepsilon \times$ *input* $\times$ *error*

     "learning rate" $(0 < \varepsilon < 1)$

target = 1

error = 1



# Perceptrons

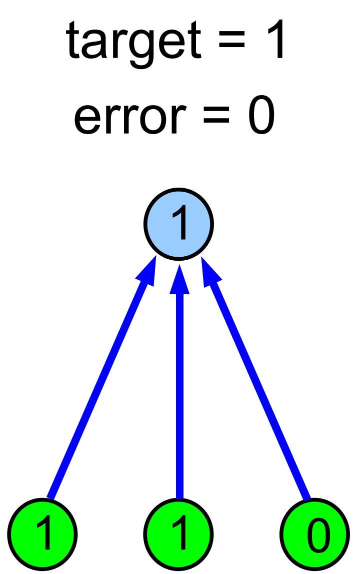
- Binary threshold neurons
- Studied by Frank Rosenblatt of Cornell in early 1960's
- Perceptron training procedure

1. present an input pattern
2. compute output value

   *output* = $\Theta$(sum of: *inputs* $\times$ *weights*)
3. compare output to target value

   *error* = *target* - *output*
4. if incorrect, adjust weights

   *weight_adjust* = $\varepsilon \times$ *input* $\times$ *error*
5. repeat until all input patterns give the correct output value



# Perceptrons

- Perceptron learning theorem

> *The perceptron training procedure is **guaranteed** to find weight values that correctly solve the problem, within a finite number of steps, provided such weight values exist.*

# Perceptrons

- Perceptron learning theorem

> *The perceptron training procedure is **guaranteed** to find weight values that correctly solve the problem, within a finite number of steps, provided such weight values exist.*

- Not all problems can be solved by single-layer perceptrons
- Classic example: XOR

$$\begin{array}{llll} 0\ 0 \Rightarrow 0 & \quad & 0\ 1 \Rightarrow 1 \\ 1\ 1 \Rightarrow 0 & \quad & 1\ 0 \Rightarrow 1 \end{array}$$

# Perceptrons

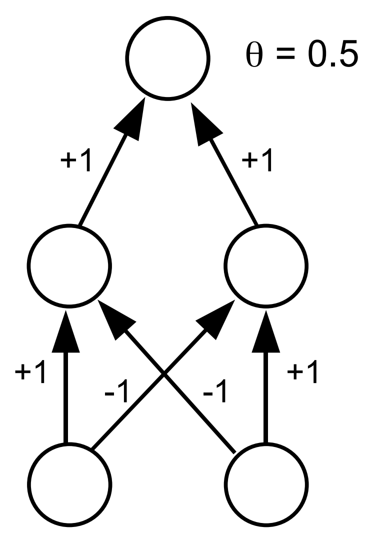
- Perceptron learning theorem

> *The perceptron training procedure is **guaranteed** to find weight values that correctly solve the problem, within a finite number of steps, provided such weight values exist.*

- Not all problems can be solved by single-layer perceptrons
- Classic example: XOR

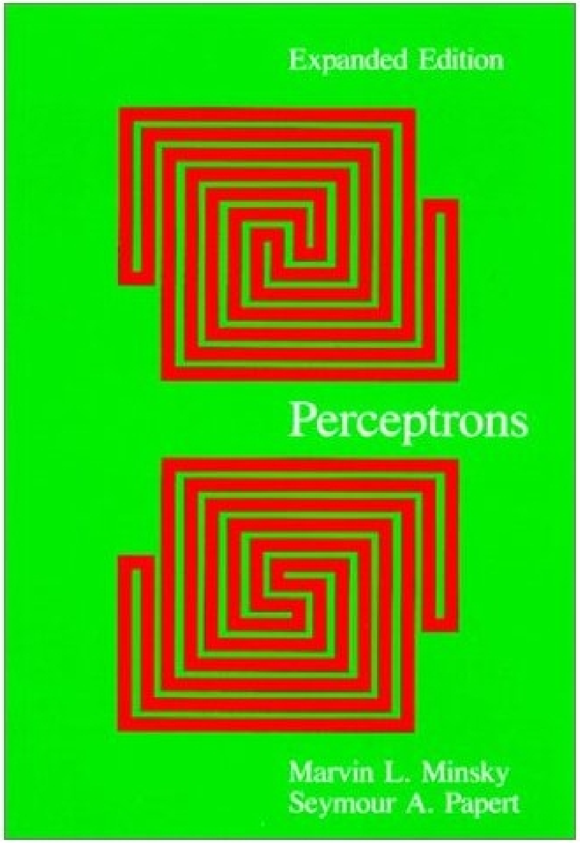
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | $\Rightarrow$ | 0 | 0 | 1 | $\Rightarrow$ | 1 |
| 1 | 1 | $\Rightarrow$ | 0 | 1 | 0 | $\Rightarrow$ | 1 |

- Perceptrons with more than one layer of weights **can** solve XOR, in principle
- No training procedure or learning theorem for multi-layer networks was known in the 1960s



# Perceptrons

- Marvin Minsky and Seymour Papert of MIT published *Perceptrons* in 1969
- They rigorously analyzed the limitations of perceptrons, and speculated that these limitations also applied to networks with multiple layers of weights
- This caused many people to seriously question the capabilities of neural networks
- As a result, interest in neural network research (and funding) largely dried up for more than a decade

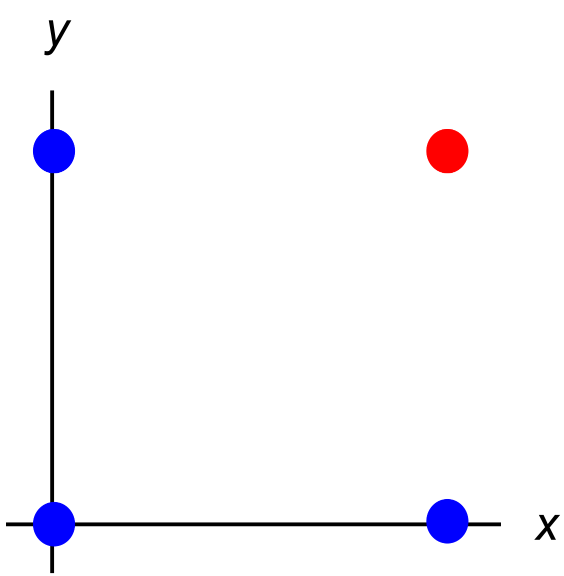


# Linear Separability

- It is helpful to think about neural networks geometrically
- Input patterns correspond to points in the **input space**
- A perceptron that correctly classifies input patterns as belonging to category A or B corresponds to a straight line dividing the input space into two halves
- The category A patterns lie in one half of the space, and the category B patterns lie in the other half
- If the input patterns can be separated by a straight line in this way, we say the problem is **linearly separable**
- Minsky and Papert proved that many interesting problems are not linearly separable, and thus no perceptron can solve them

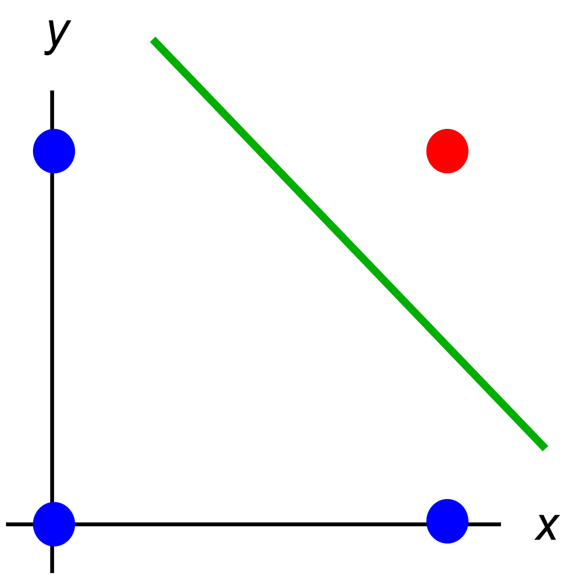
# Linear Separability

| *x* | *y* | *x* **AND** *y* |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



# Linear Separability

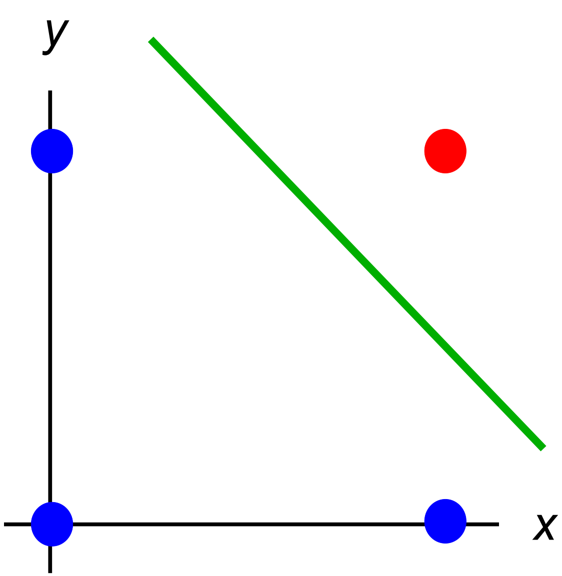
| $x$ | $y$ | $x$ **AND** $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

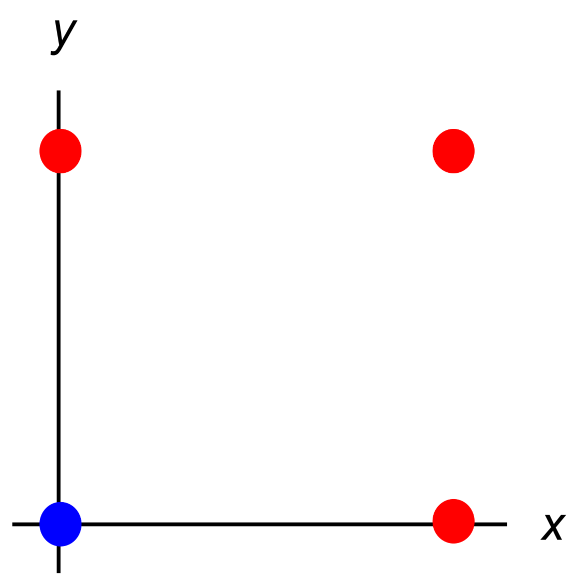


# Linear Separability

| $x$ | $y$ | $x$ **AND** $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| $x$ | $y$ | $x$ **OR** $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

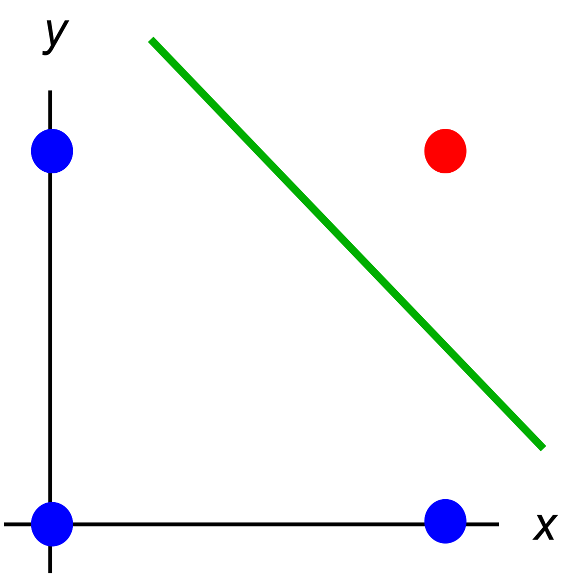


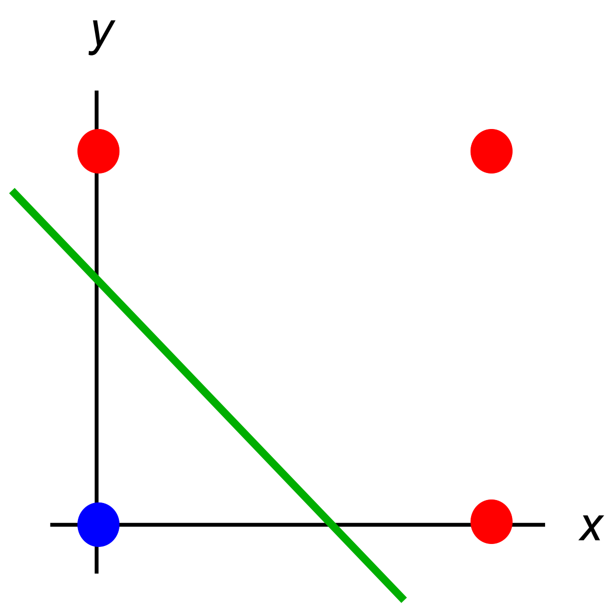



# Linear Separability

| *x* | *y* | *x* **AND** *y* |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| *x* | *y* | *x* **OR** *y* |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |



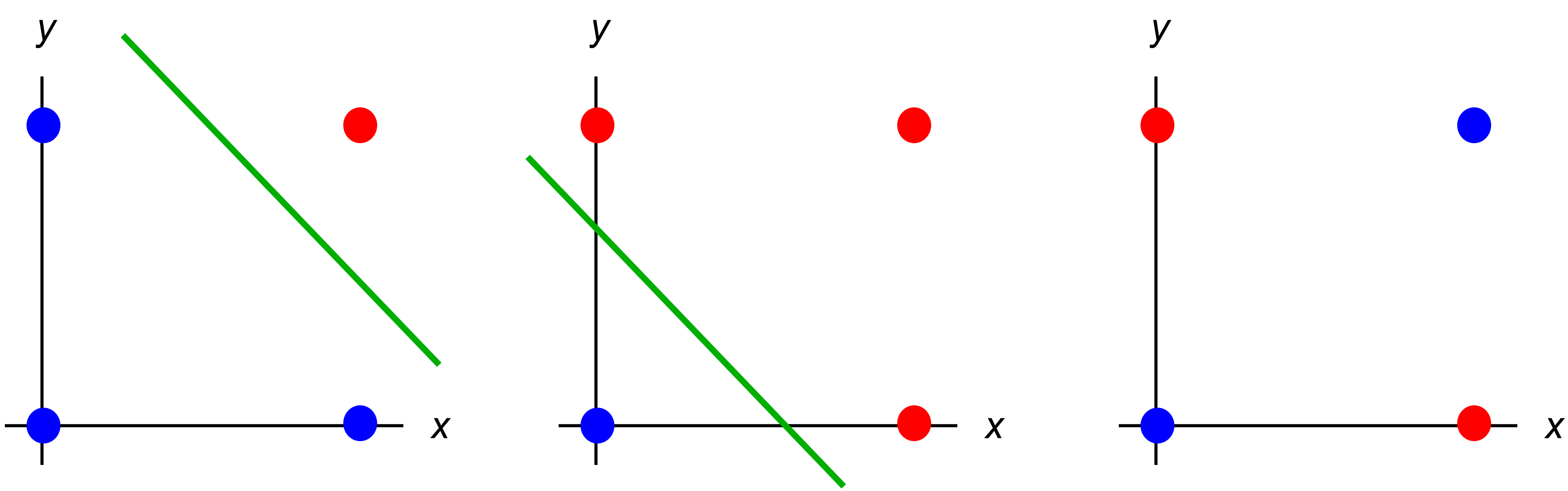



# Linear Separability

| $x$ | $y$ | $x$ **AND** $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| $x$ | $y$ | $x$ **OR** $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| $x$ | $y$ | $x$ **XOR** $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

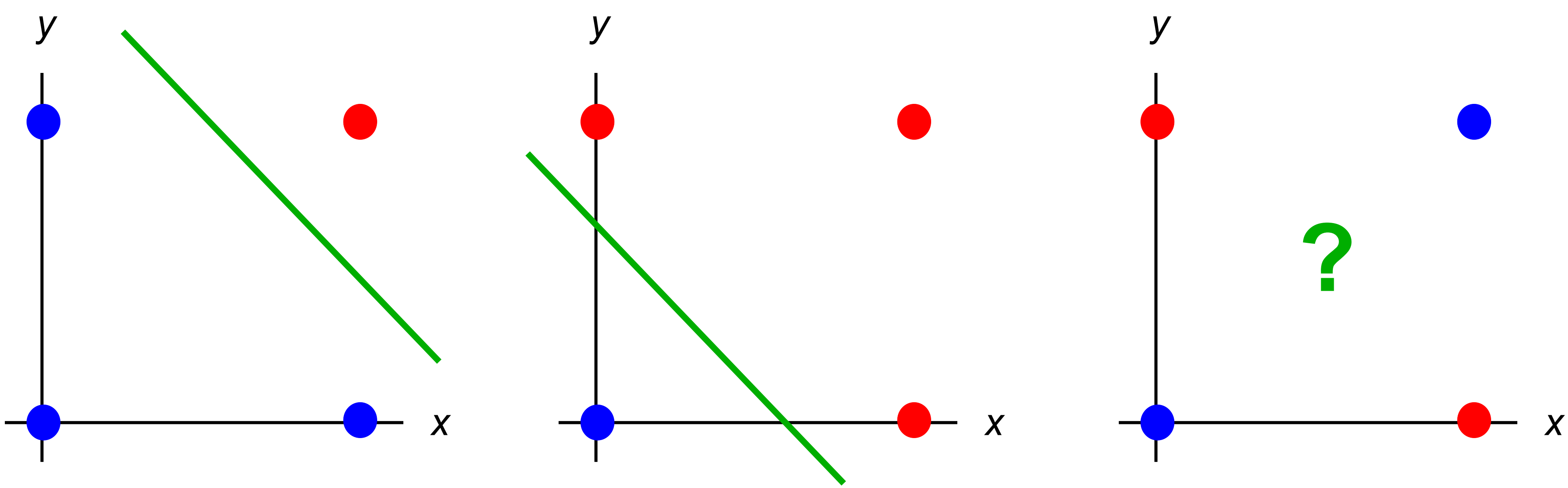


# Linear Separability

| *x* | *y* | *x* **AND** *y* |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

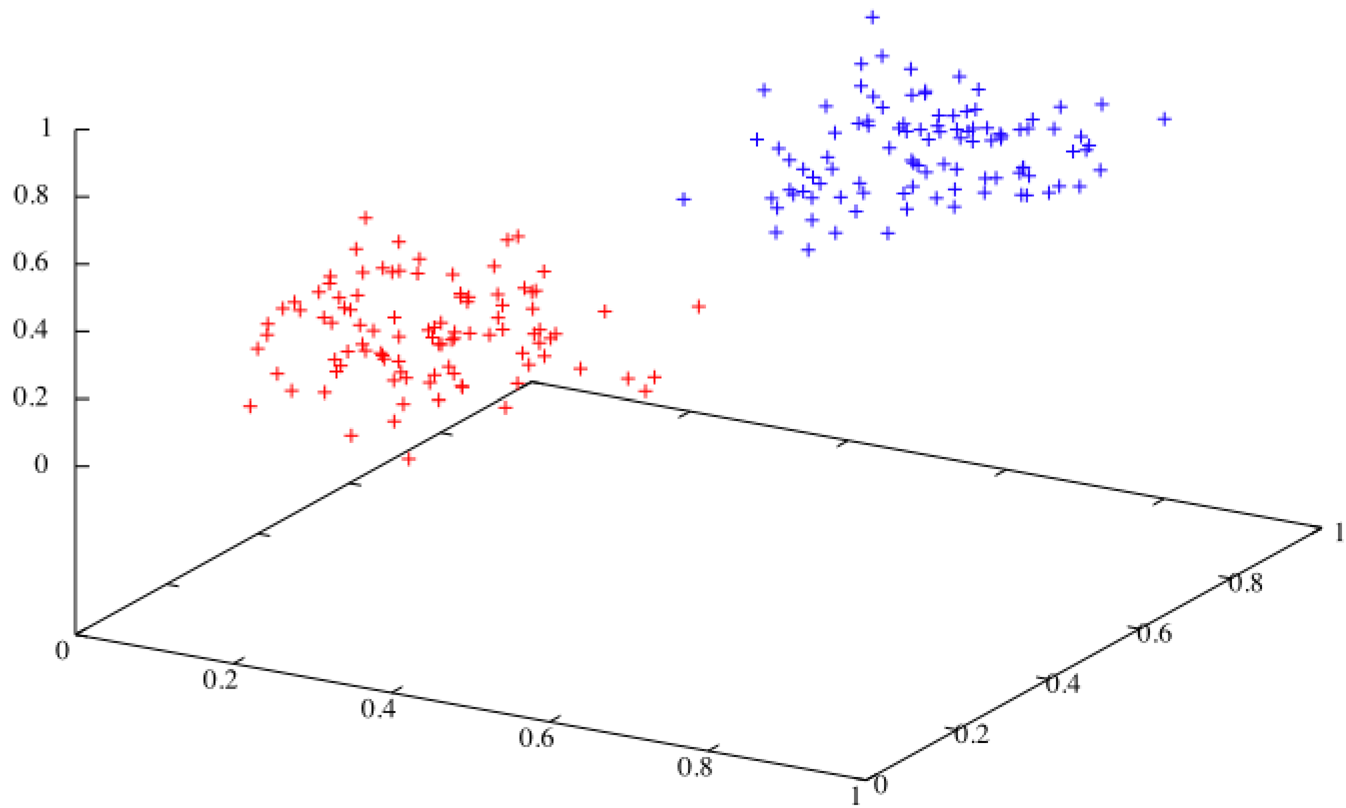
| *x* | *y* | *x* **OR** *y* |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| *x* | *y* | *x* **XOR** *y* |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

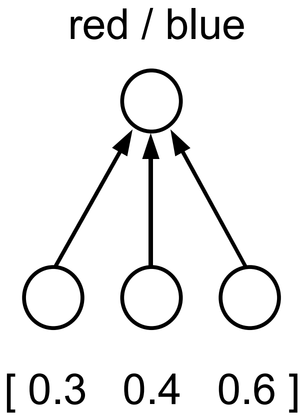


# Linear Separability

- This idea applies to input spaces of any dimensionality
- Example: 3-dimensional input patterns

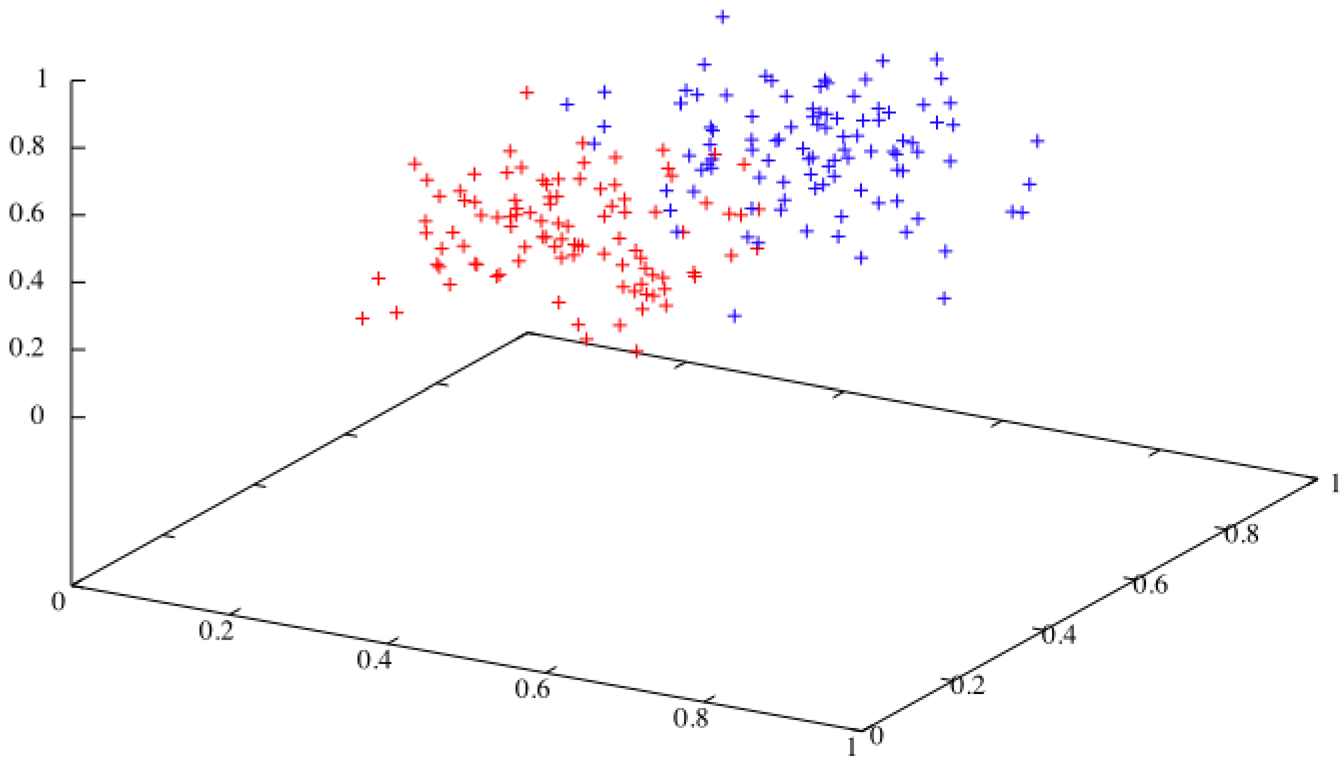


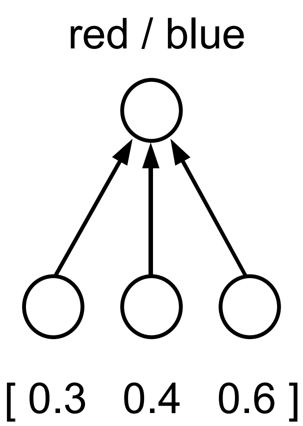

linearly separable



# Linear Separability

- This idea applies to input spaces of any dimensionality
- Example: 3-dimensional input patterns

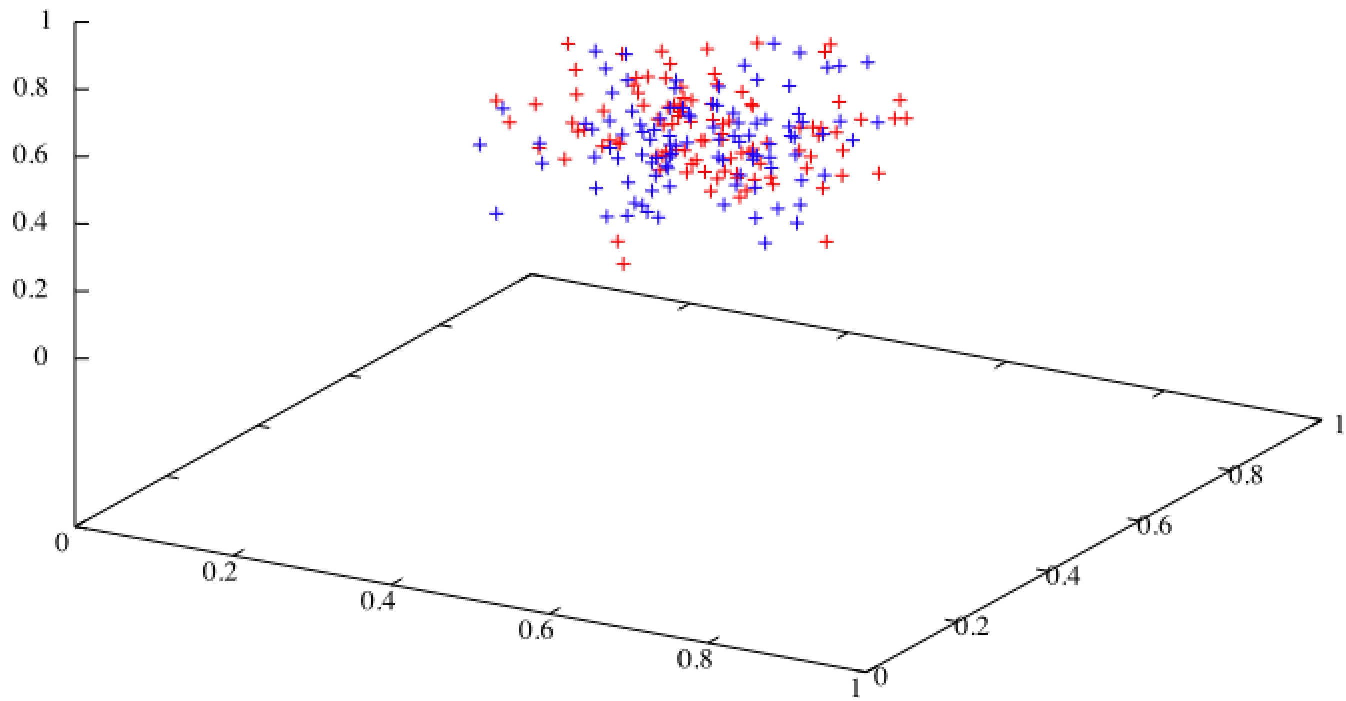


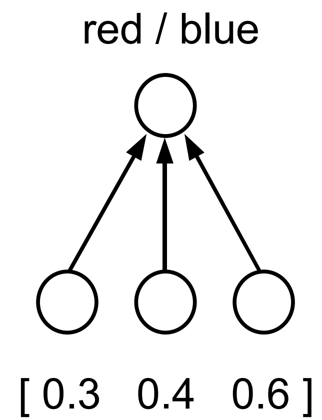




partially linearly separable

# Linear Separability

- This idea applies to input spaces of any dimensionality
- Example: 3-dimensional input patterns

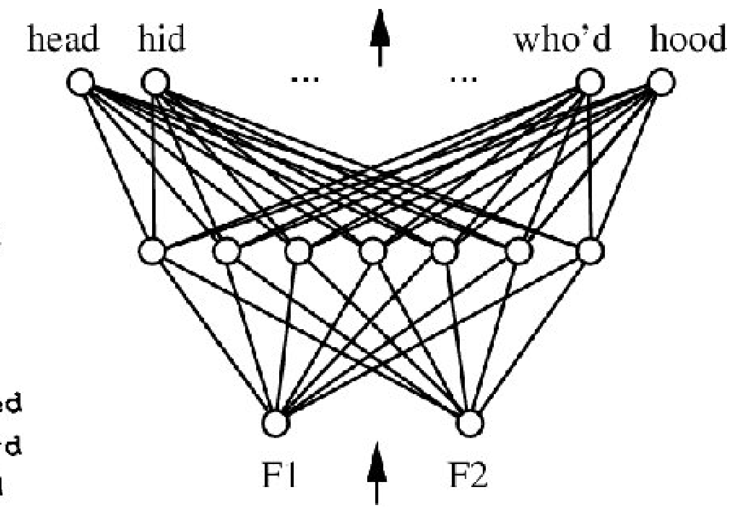


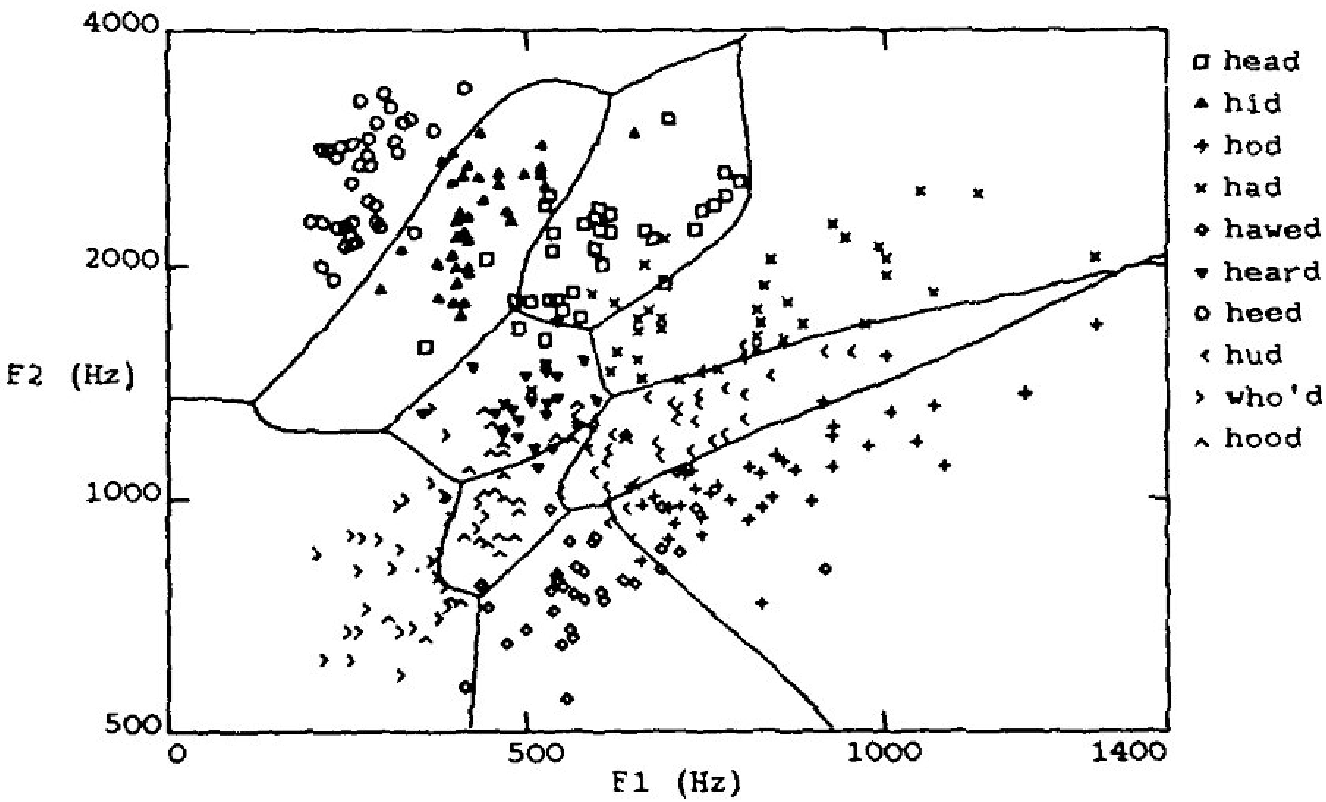




not linearly separable

# Linear Separability

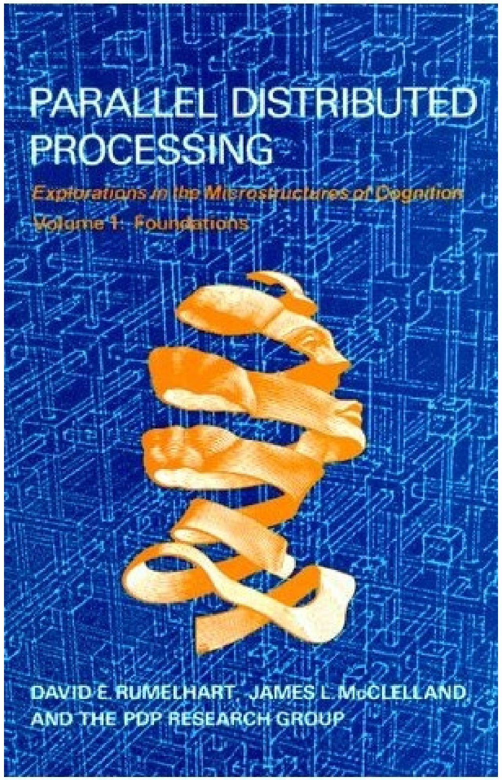
- Multi-layer networks can learn to classify input patterns that are not linearly separable
- Example: recognizing vowels

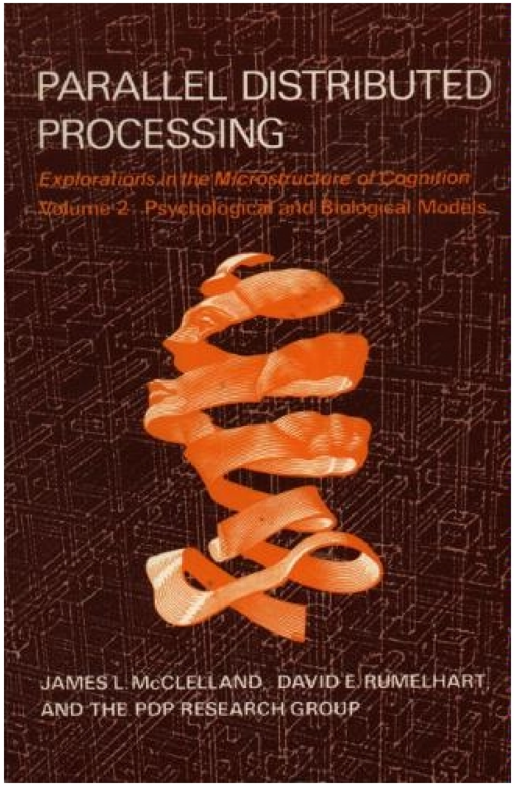


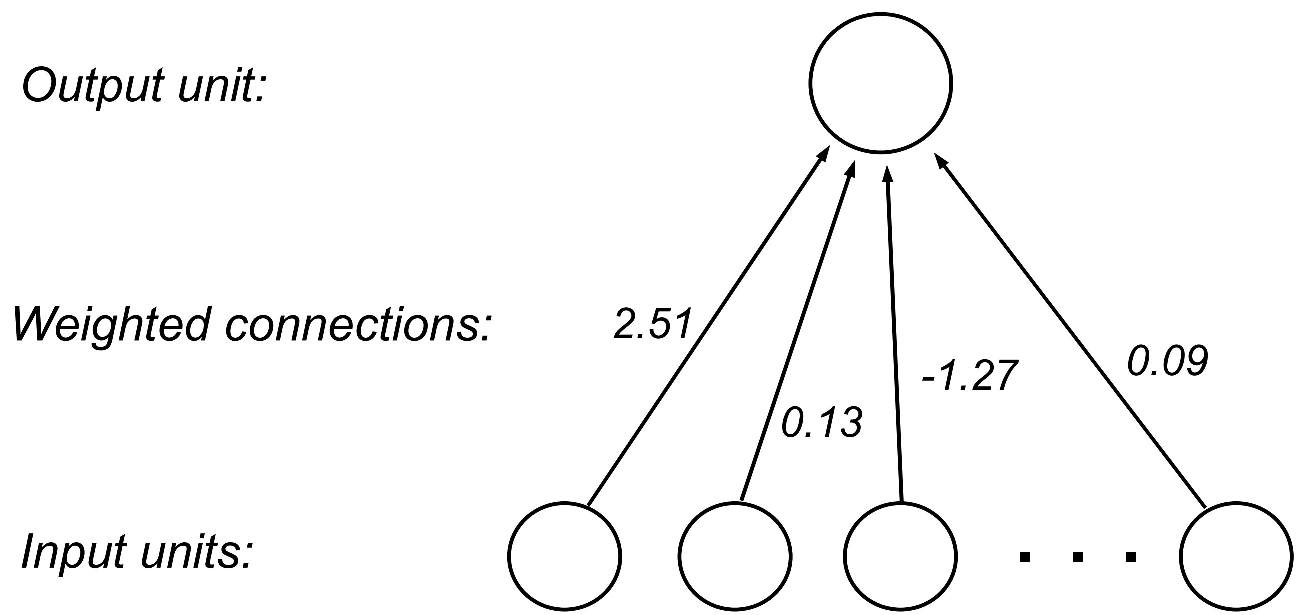
# Parallel Distributed Processing (PDP)

- In the 1980s, a way to train multi-layer networks was discovered, called the **backpropagation** learning algorithm
- David Rumelhart, Geoffrey Hinton, James McClelland, and others revived interest in neural networks with the publication of the "PDP books"
- John Hopfield analyzed networks that behaved as memories, using techniques from physics
- The field has been going strong ever since

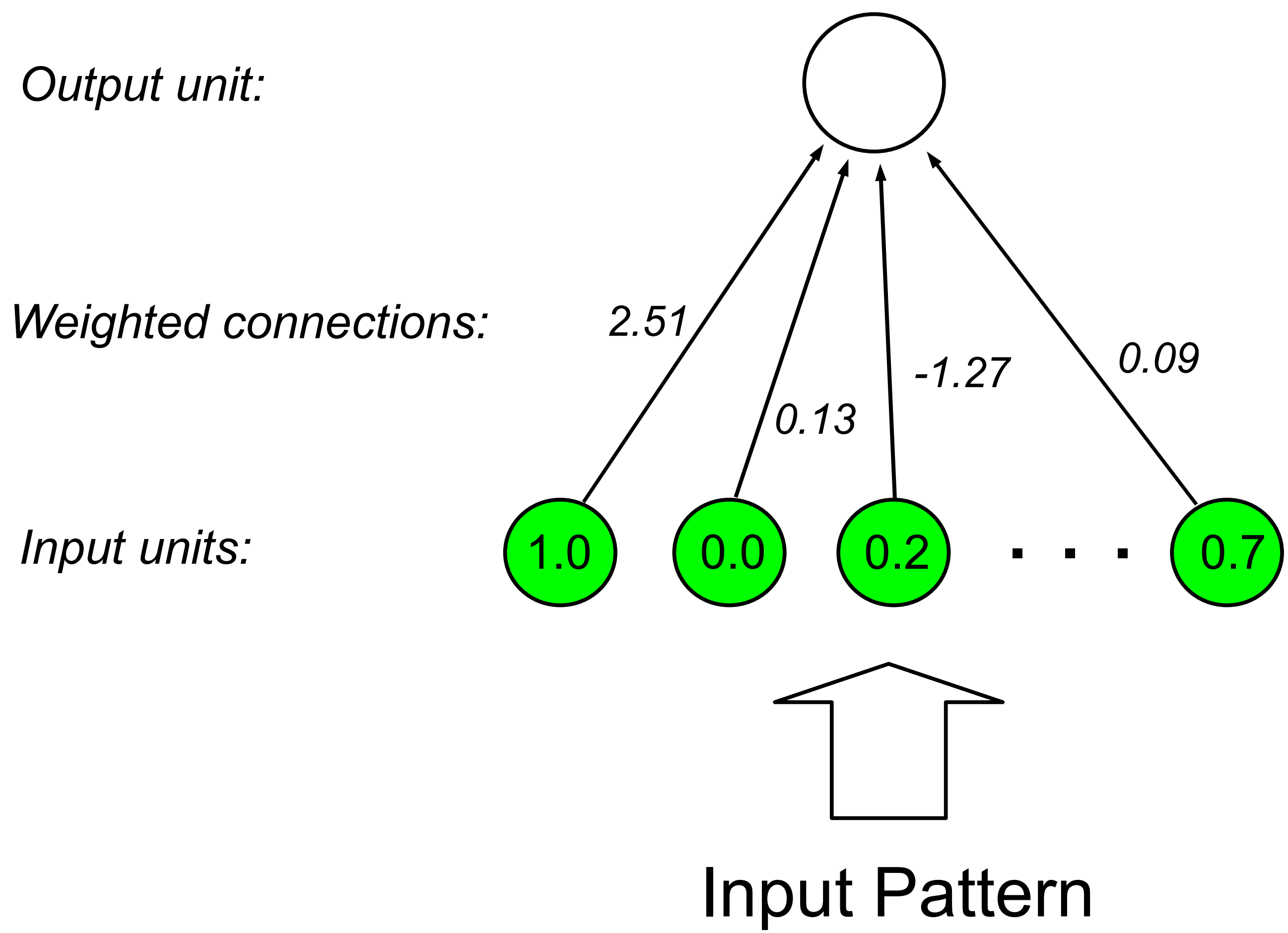


# Artificial Neurons: Continuous Version

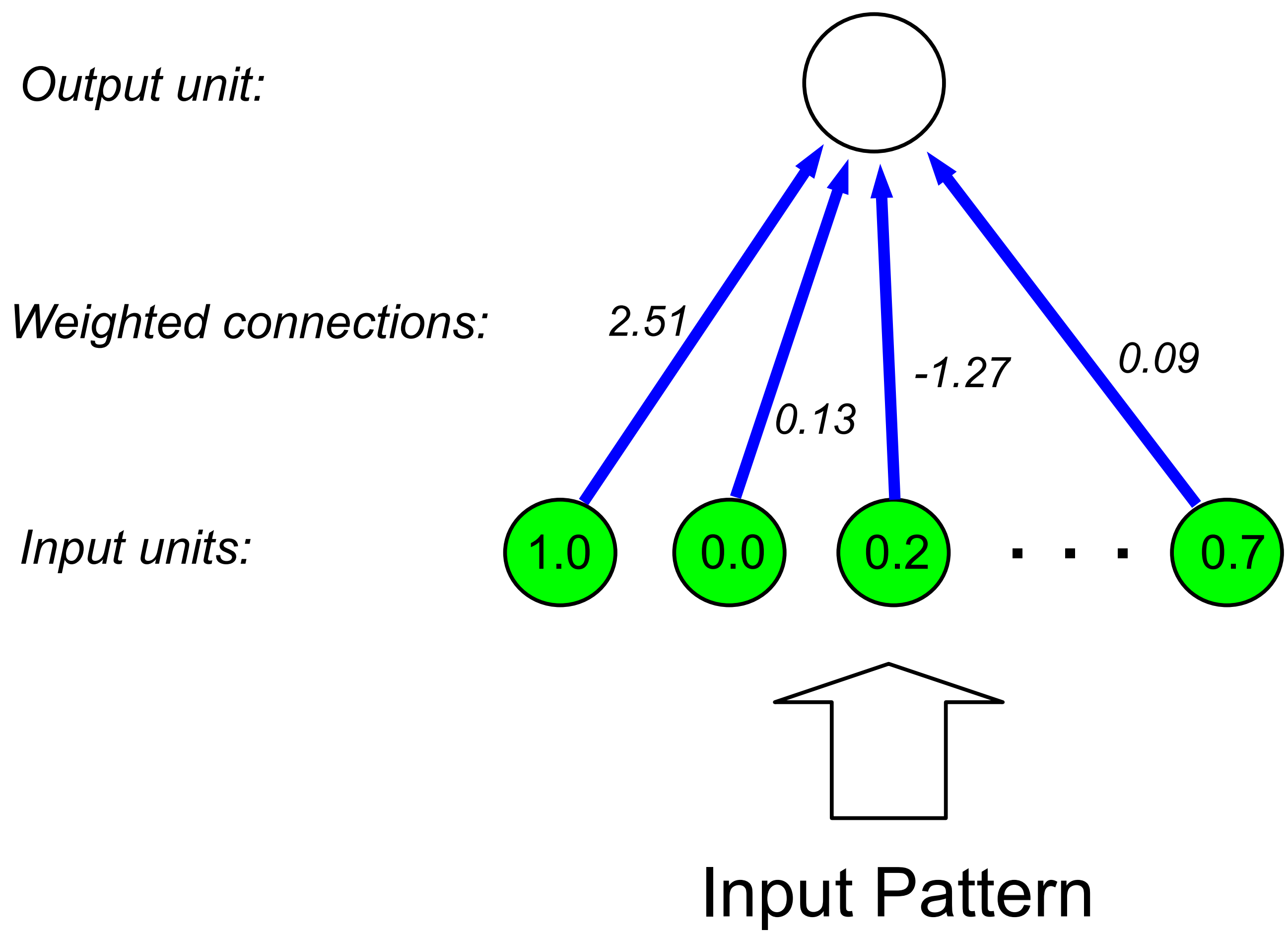


# Artificial Neurons: Continuous Version



# Artificial Neurons: Continuous Version

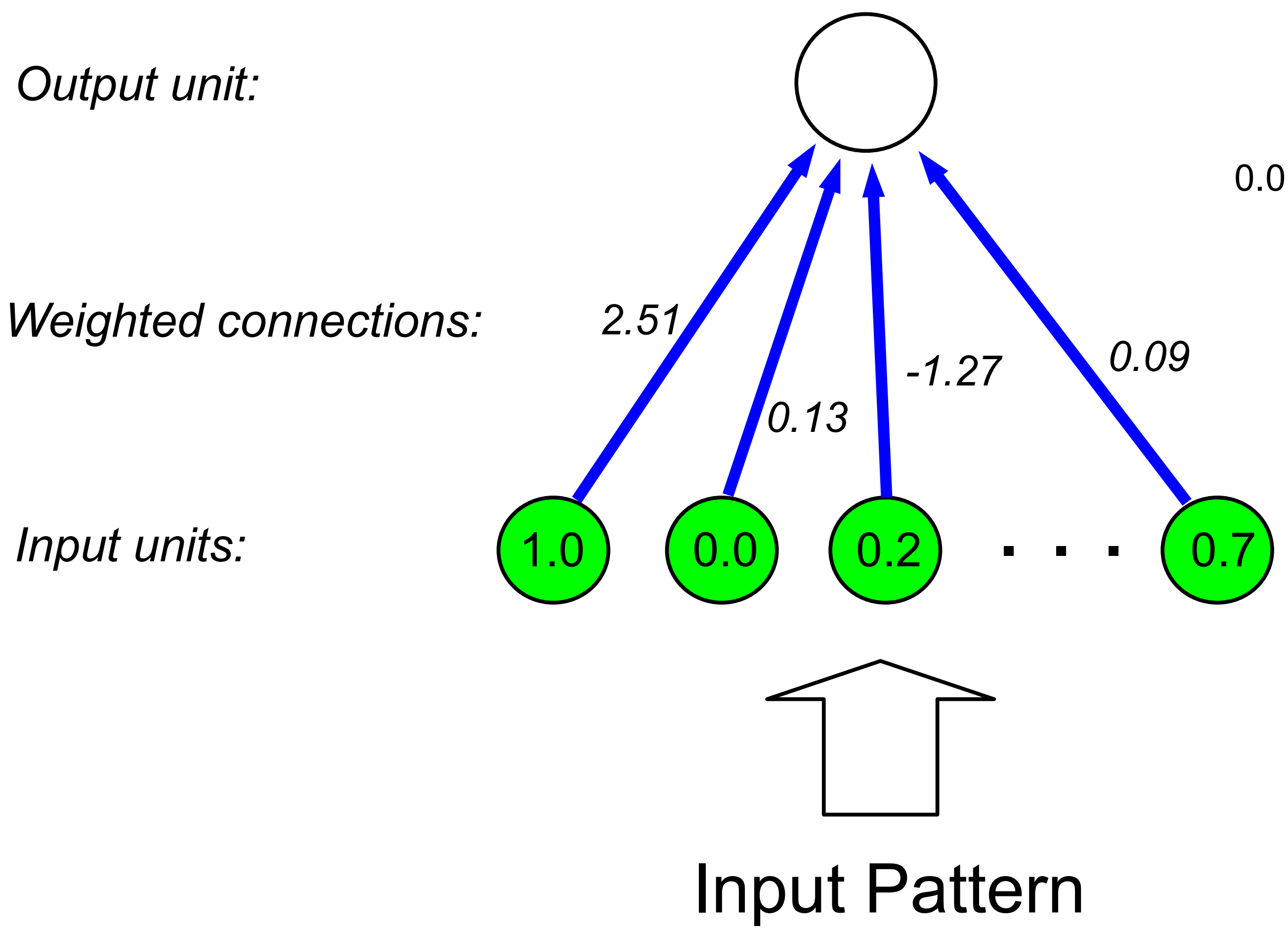
$$\mathbf{1.0} \times 2.51 \; + \; \mathbf{0.0} \times 0.13 \; + \; \mathbf{0.2} \times -1.27 \; + \; \mathbf{0.7} \times 0.09 \; = \; 2.32$$

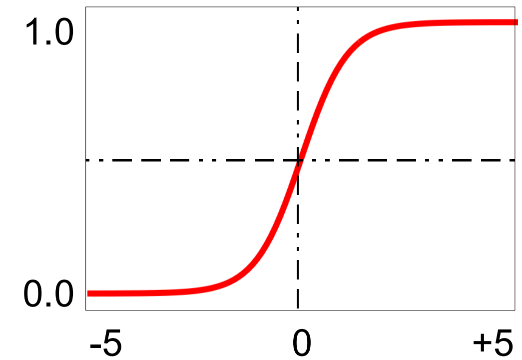


# Artificial Neurons: Continuous Version

$$\mathbf{1.0} \times 2.51 + \mathbf{0.0} \times 0.13 + \mathbf{0.2} \times -1.27 + \mathbf{0.7} \times 0.09 = 2.32$$

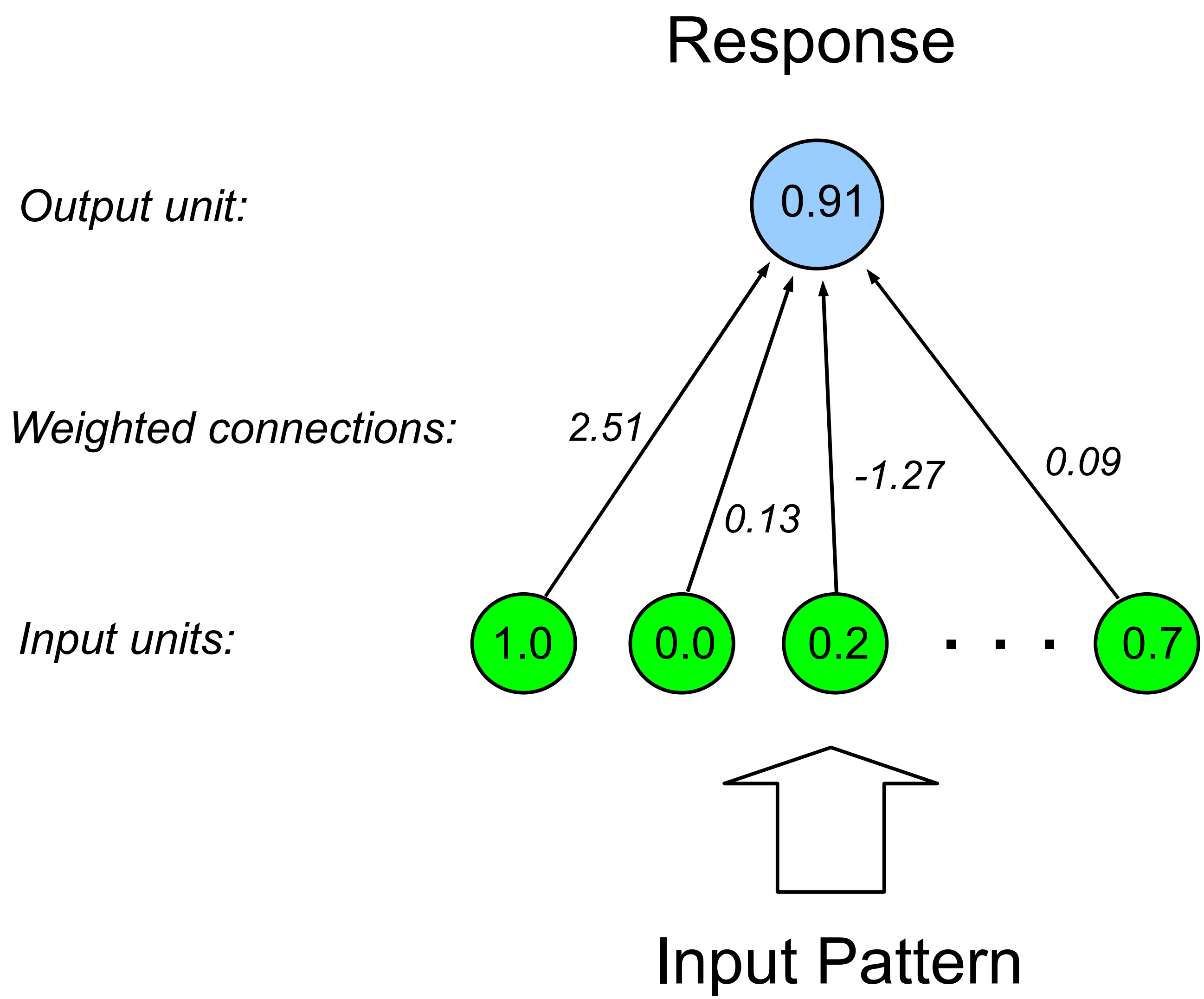
$$\sigma(2.32) = 0.91$$




$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

# Artificial Neurons: Continuous Version



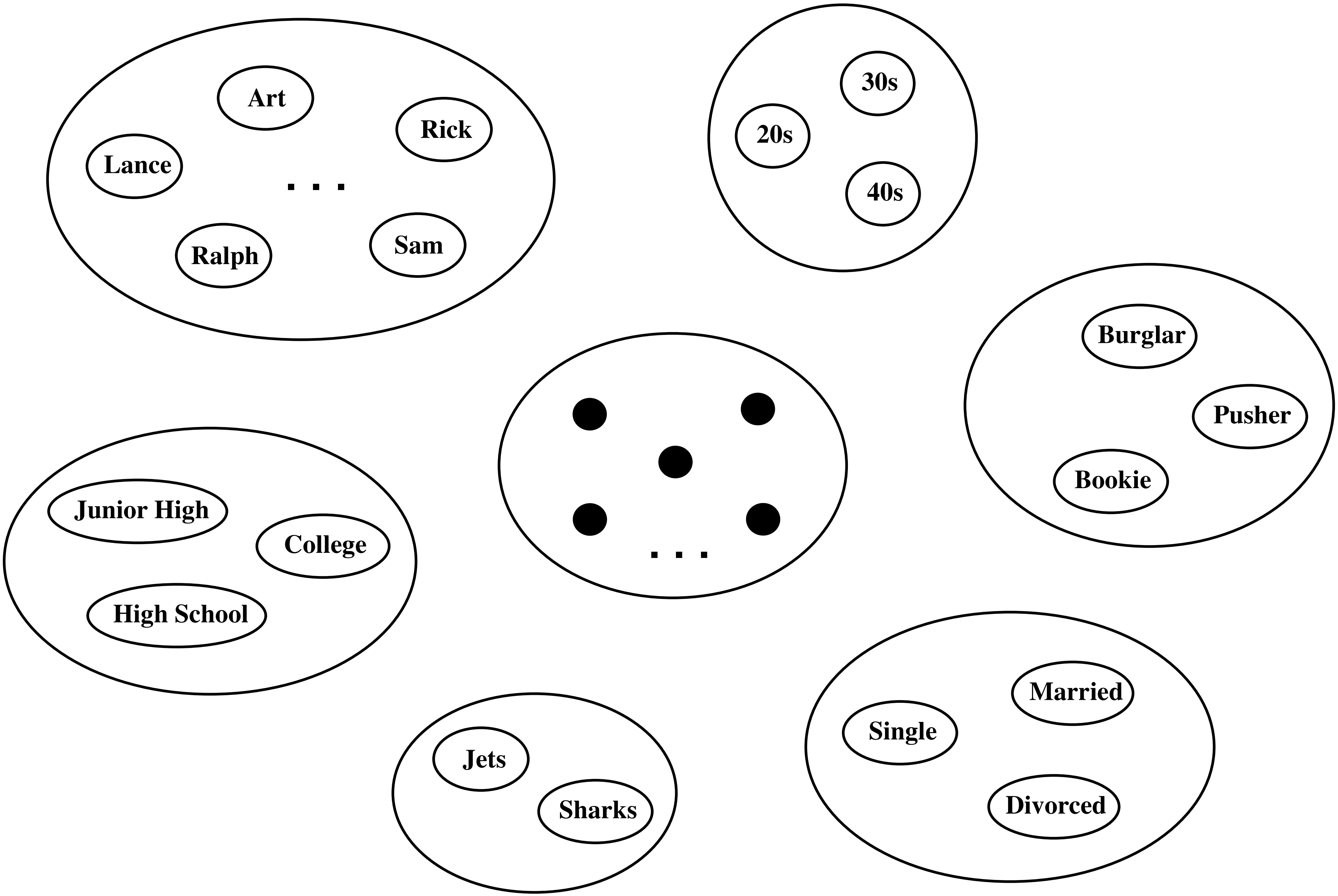
# Constraint Satisfaction Networks

- No learning occurs
- Connection strengths are permanently fixed
- Excitatory and inhibitory feedback connections
- Node activations represent current state of the network
- Node activations settle into a stable pattern over time
- Can behave as a content-addressable memory
- Examples:
  - Jets and Sharks network
  - Hopfield memories

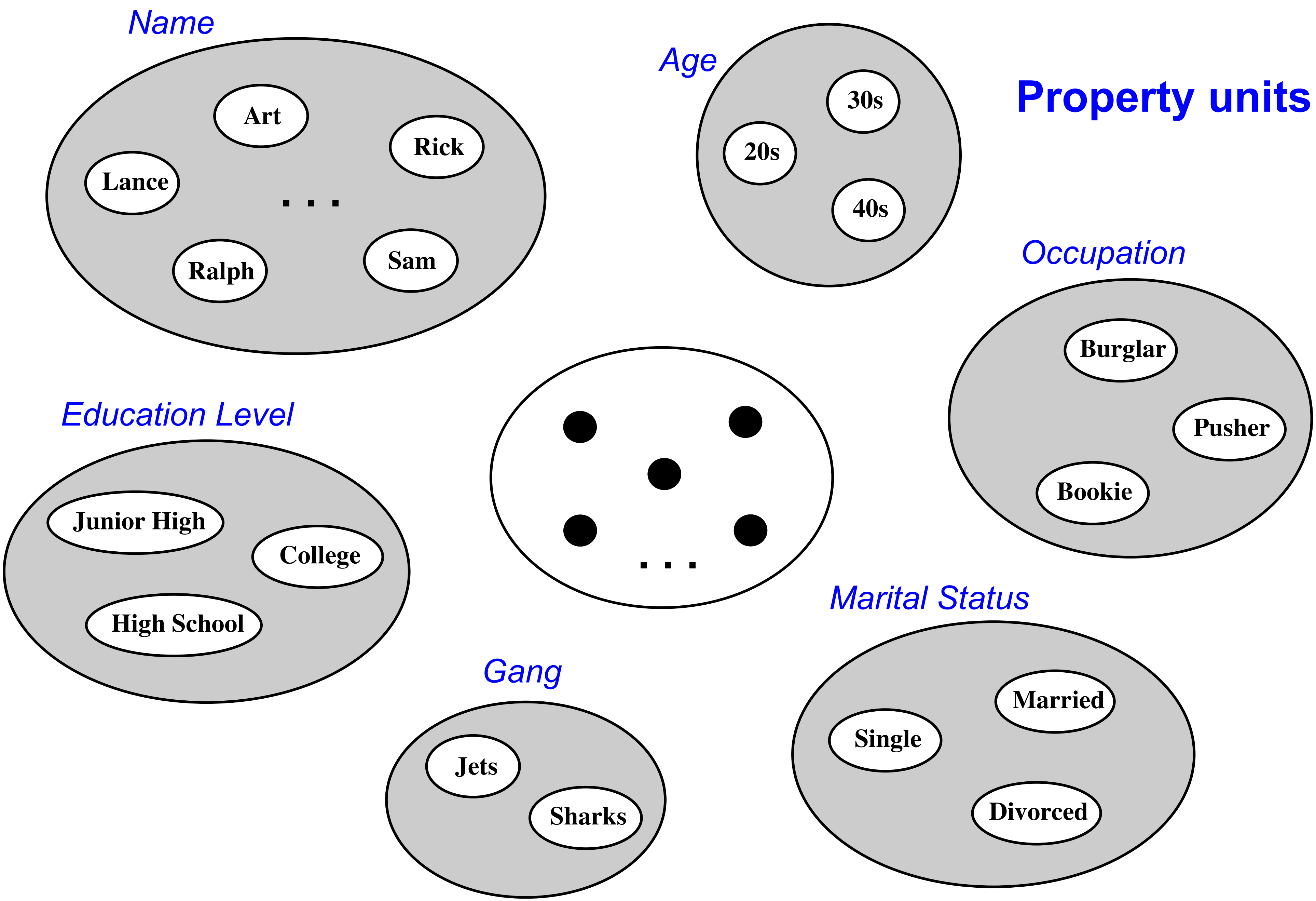
# The Jets and the Sharks

| *Name* | *Gang* | *Age* | *Education* | *Marital Status* | *Occupation* |
|---|---|---|---|---|---|
| **Art** | **Jets** | **40s** | **Junior High** | **Single** | **Pusher** |
| **Al** | **Jets** | **30s** | **Junior High** | **Married** | **Burglar** |
| **Sam** | **Jets** | **20s** | **College** | **Single** | **Bookie** |
| **Clyde** | **Jets** | **40s** | **Junior High** | **Single** | **Bookie** |
| **Mike** | **Jets** | **30s** | **Junior High** | **Single** | **Bookie** |
| **Jim** | **Jets** | **20s** | **Junior High** | **Divorced** | **Burglar** |
| **Greg** | **Jets** | **20s** | **High School** | **Married** | **Pusher** |
| **John** | **Jets** | **20s** | **Junior High** | **Married** | **Burglar** |
| **Doug** | **Jets** | **30s** | **High School** | **Single** | **Bookie** |
| **Lance** | **Jets** | **20s** | **Junior High** | **Married** | **Burglar** |
| **George** | **Jets** | **20s** | **Junior High** | **Divorced** | **Burglar** |
| **Pete** | **Jets** | **20s** | **High School** | **Single** | **Bookie** |
| **Fred** | **Jets** | **20s** | **High School** | **Single** | **Pusher** |
| **Gene** | **Jets** | **20s** | **College** | **Single** | **Pusher** |
| **Ralph** | **Jets** | **30s** | **Junior High** | **Single** | **Pusher** |
| **Phil** | **Sharks** | **30s** | **College** | **Married** | **Pusher** |
| **Ike** | **Sharks** | **30s** | **Junior High** | **Single** | **Bookie** |
| **Nick** | **Sharks** | **30s** | **High School** | **Single** | **Pusher** |
| **Don** | **Sharks** | **30s** | **College** | **Married** | **Burglar** |
| **Ned** | **Sharks** | **30s** | **College** | **Married** | **Bookie** |
| **Karl** | **Sharks** | **40s** | **High School** | **Married** | **Bookie** |
| **Ken** | **Sharks** | **20s** | **High School** | **Single** | **Burglar** |
| **Earl** | **Sharks** | **40s** | **High School** | **Married** | **Burglar** |
| **Rick** | **Sharks** | **30s** | **High School** | **Divorced** | **Burglar** |
| **Ol** | **Sharks** | **30s** | **College** | **Married** | **Pusher** |
| **Neal** | **Sharks** | **30s** | **High School** | **Single** | **Bookie** |
| **Dave** | **Sharks** | **30s** | **High School** | **Divorced** | **Pusher** |

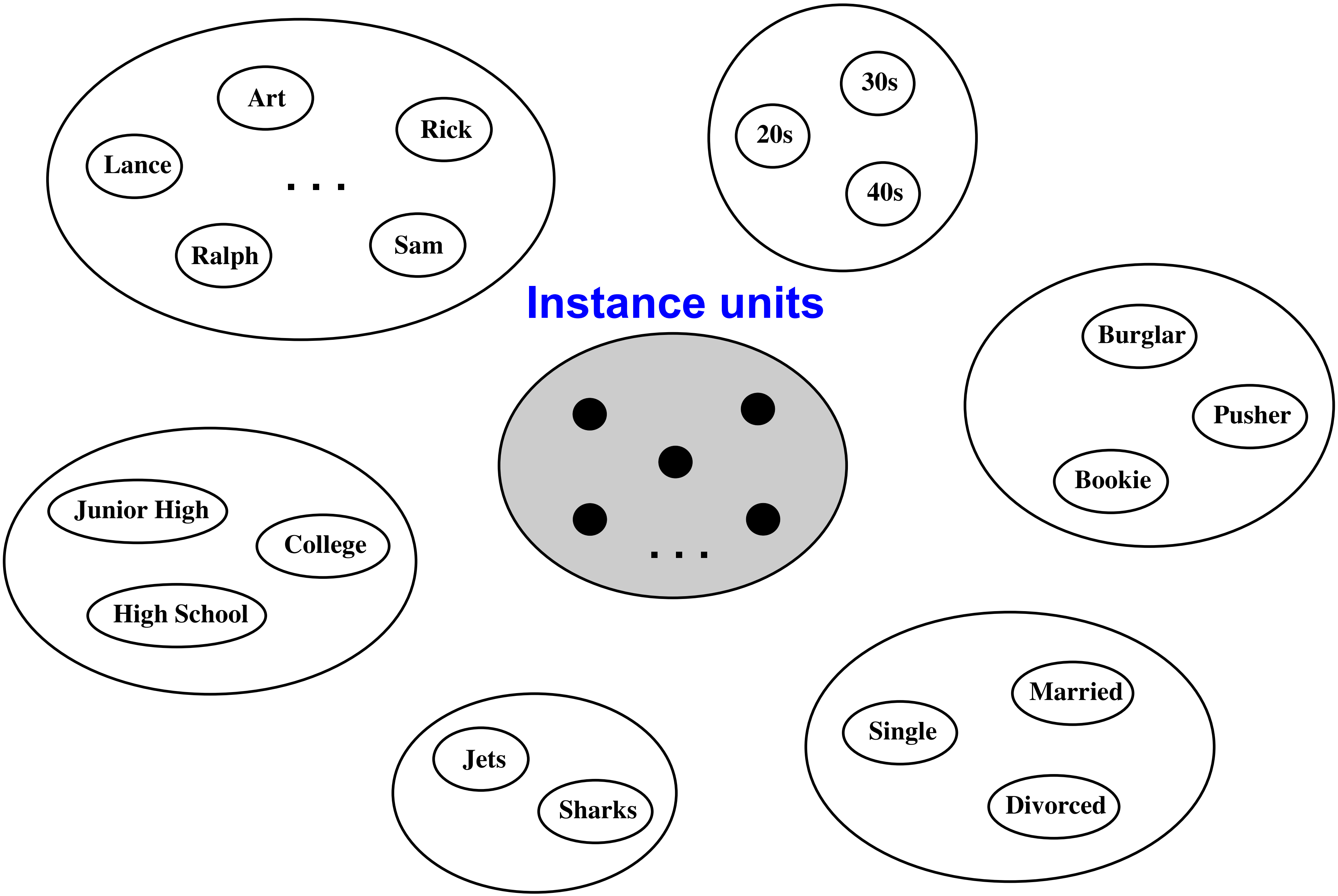
# The Jets and the Sharks



# The Jets and the Sharks



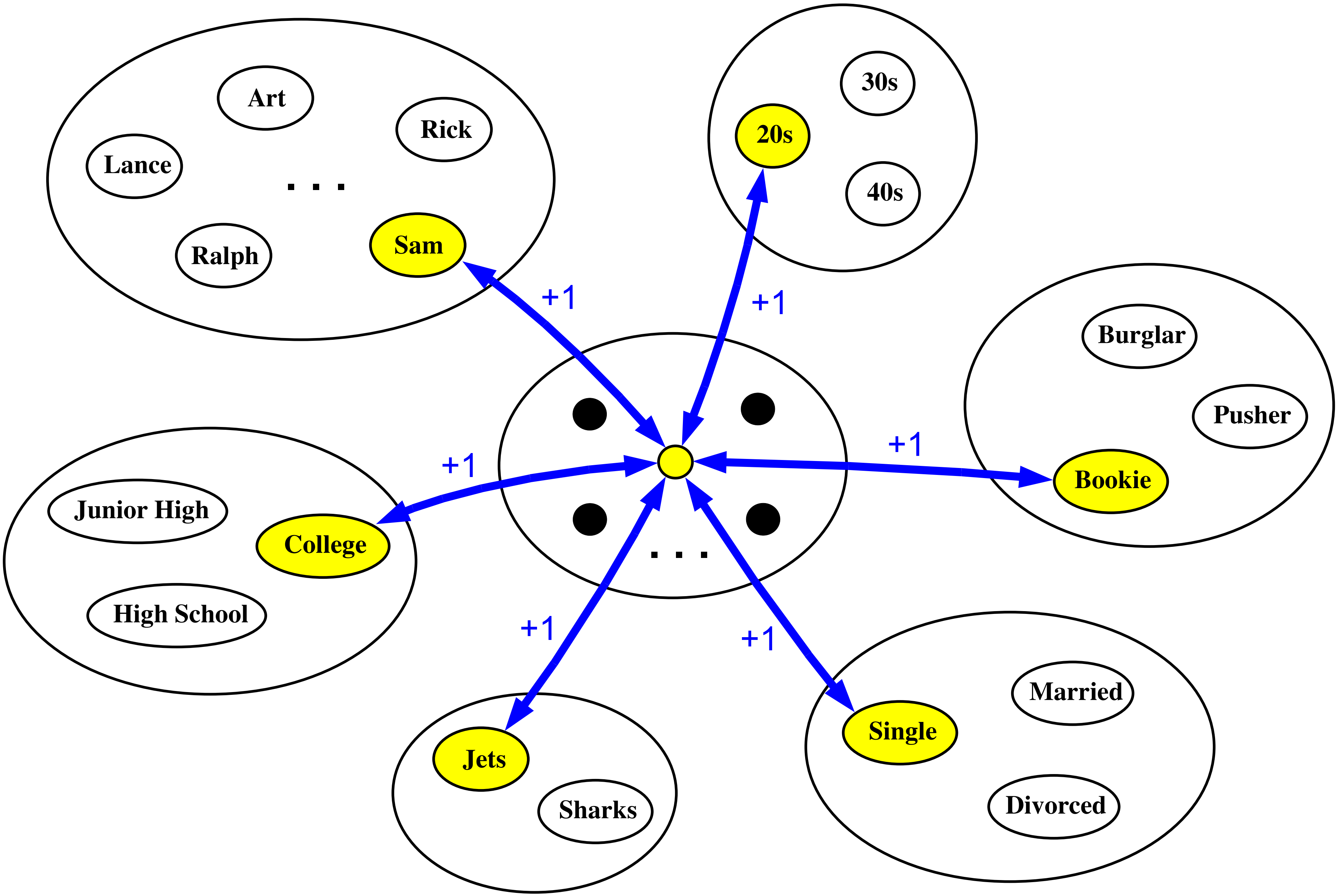
# The Jets and the Sharks



# The Jets and the Sharks

| *Name* | *Gang* | *Age* | *Education* | *Marital Status* | *Occupation* |
|---|---|---|---|---|---|
| **Art** | **Jets** | **40s** | **Junior High** | **Single** | **Pusher** |
| **Al** | **Jets** | **30s** | **Junior High** | **Married** | **Burglar** |
| **Sam** | **Jets** | **20s** | **College** | **Single** | **Bookie** |
| **Clyde** | **Jets** | **40s** | **Junior High** | **Single** | **Bookie** |
| **Mike** | **Jets** | **30s** | **Junior High** | **Single** | **Bookie** |
| **Jim** | **Jets** | **20s** | **Junior High** | **Divorced** | **Burglar** |
| **Greg** | **Jets** | **20s** | **High School** | **Married** | **Pusher** |
| **John** | **Jets** | **20s** | **Junior High** | **Married** | **Burglar** |
| **Doug** | **Jets** | **30s** | **High School** | **Single** | **Bookie** |
| **Lance** | **Jets** | **20s** | **Junior High** | **Married** | **Burglar** |
| **George** | **Jets** | **20s** | **Junior High** | **Divorced** | **Burglar** |
| **Pete** | **Jets** | **20s** | **High School** | **Single** | **Bookie** |
| **Fred** | **Jets** | **20s** | **High School** | **Single** | **Pusher** |
| **Gene** | **Jets** | **20s** | **College** | **Single** | **Pusher** |
| **Ralph** | **Jets** | **30s** | **Junior High** | **Single** | **Pusher** |
| **Phil** | **Sharks** | **30s** | **College** | **Married** | **Pusher** |
| **Ike** | **Sharks** | **30s** | **Junior High** | **Single** | **Bookie** |
| **Nick** | **Sharks** | **30s** | **High School** | **Single** | **Pusher** |
| **Don** | **Sharks** | **30s** | **College** | **Married** | **Burglar** |
| **Ned** | **Sharks** | **30s** | **College** | **Married** | **Bookie** |
| **Karl** | **Sharks** | **40s** | **High School** | **Married** | **Bookie** |
| **Ken** | **Sharks** | **20s** | **High School** | **Single** | **Burglar** |
| **Earl** | **Sharks** | **40s** | **High School** | **Married** | **Burglar** |
| **Rick** | **Sharks** | **30s** | **High School** | **Divorced** | **Burglar** |
| **Ol** | **Sharks** | **30s** | **College** | **Married** | **Pusher** |
| **Neal** | **Sharks** | **30s** | **High School** | **Single** | **Bookie** |
| **Dave** | **Sharks** | **30s** | **High School** | **Divorced** | **Pusher** |

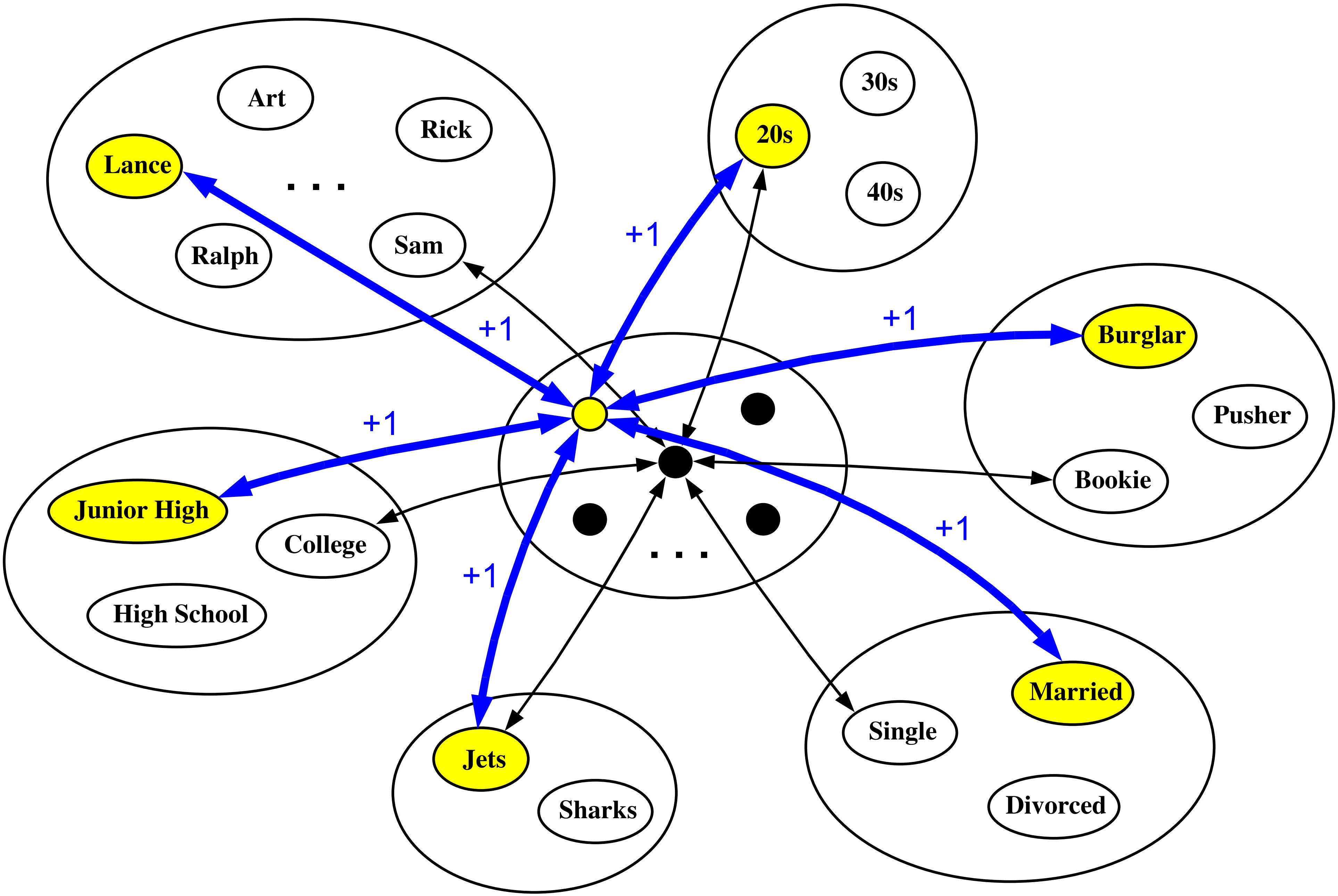
# The Jets and the Sharks



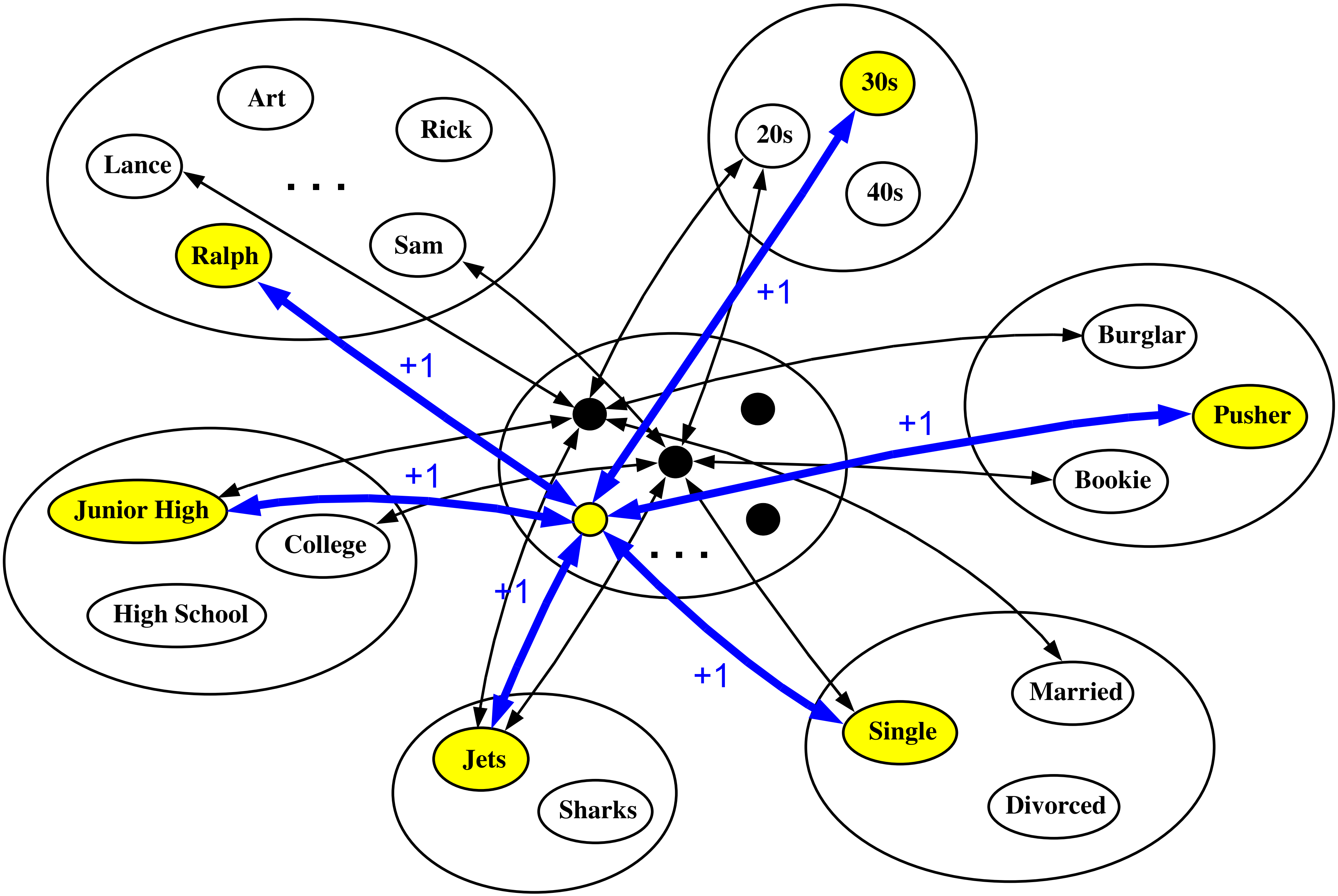
# The Jets and the Sharks

| *Name* | *Gang* | *Age* | *Education* | *Marital Status* | *Occupation* |
|---|---|---|---|---|---|
| Art | Jets | 40s | Junior High | Single | Pusher |
| Al | Jets | 30s | Junior High | Married | Burglar |
| Sam | Jets | 20s | College | Single | Bookie |
| Clyde | Jets | 40s | Junior High | Single | Bookie |
| Mike | Jets | 30s | Junior High | Single | Bookie |
| Jim | Jets | 20s | Junior High | Divorced | Burglar |
| Greg | Jets | 20s | High School | Married | Pusher |
| John | Jets | 20s | Junior High | Married | Burglar |
| Doug | Jets | 30s | High School | Single | Bookie |
| Lance | Jets | 20s | Junior High | Married | Burglar |
| George | Jets | 20s | Junior High | Divorced | Burglar |
| Pete | Jets | 20s | High School | Single | Bookie |
| Fred | Jets | 20s | High School | Single | Pusher |
| Gene | Jets | 20s | College | Single | Pusher |
| Ralph | Jets | 30s | Junior High | Single | Pusher |
| Phil | Sharks | 30s | College | Married | Pusher |
| Ike | Sharks | 30s | Junior High | Single | Bookie |
| Nick | Sharks | 30s | High School | Single | Pusher |
| Don | Sharks | 30s | College | Married | Burglar |
| Ned | Sharks | 30s | College | Married | Bookie |
| Karl | Sharks | 40s | High School | Married | Bookie |
| Ken | Sharks | 20s | High School | Single | Burglar |
| Earl | Sharks | 40s | High School | Married | Burglar |
| Rick | Sharks | 30s | High School | Divorced | Burglar |
| Ol | Sharks | 30s | College | Married | Pusher |
| Neal | Sharks | 30s | High School | Single | Bookie |
| Dave | Sharks | 30s | High School | Divorced | Pusher |

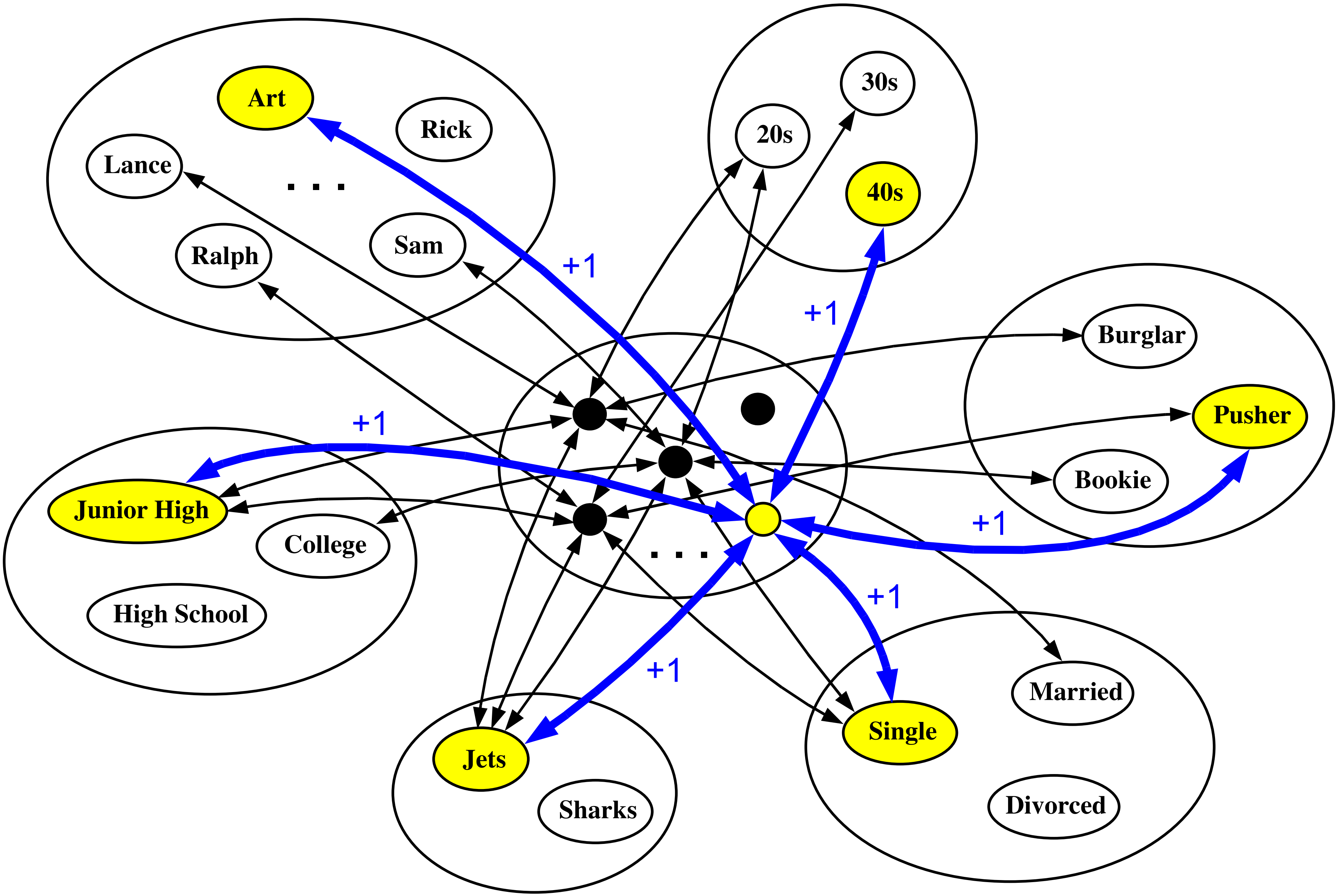
# The Jets and the Sharks



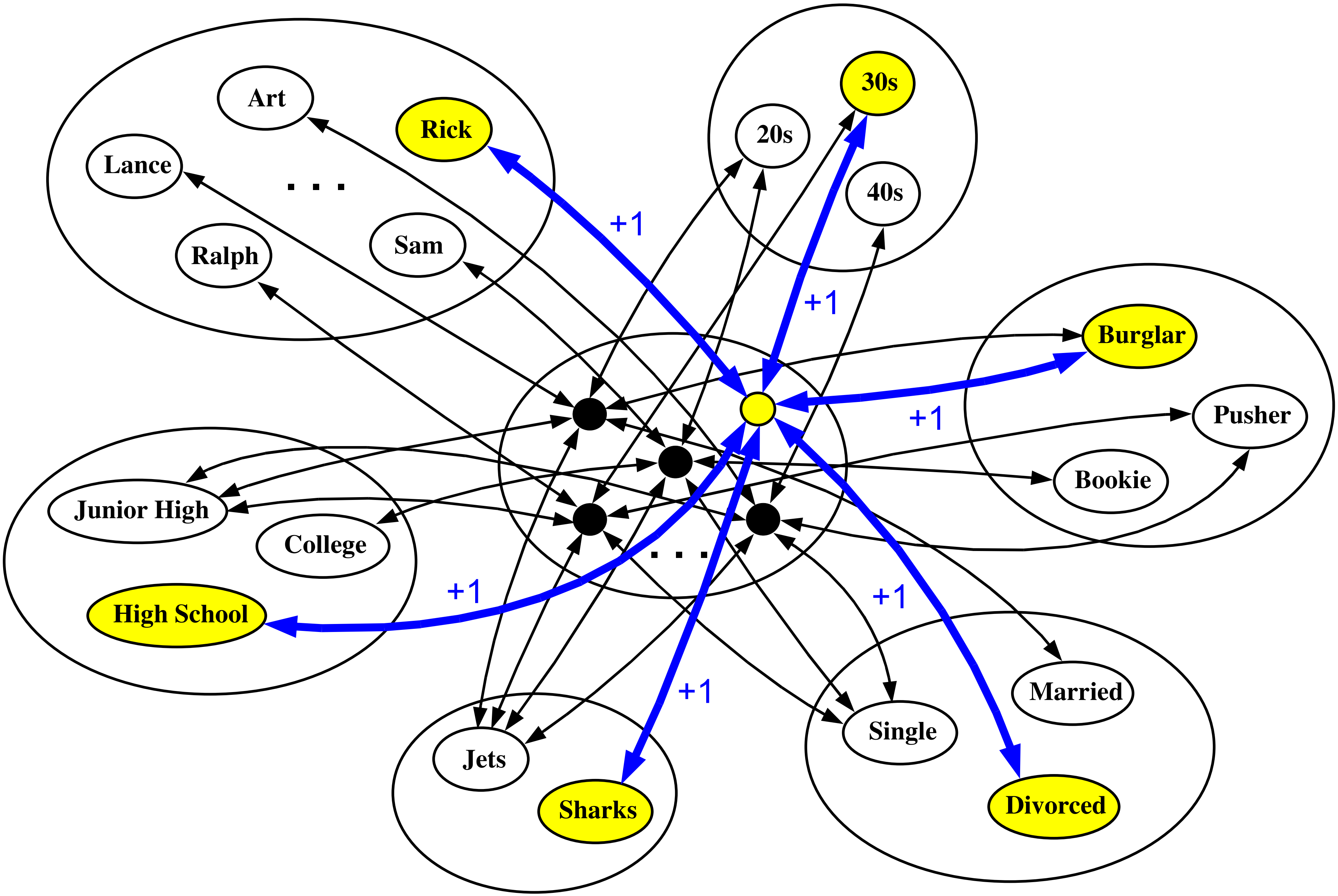
# The Jets and the Sharks



# The Jets and the Sharks

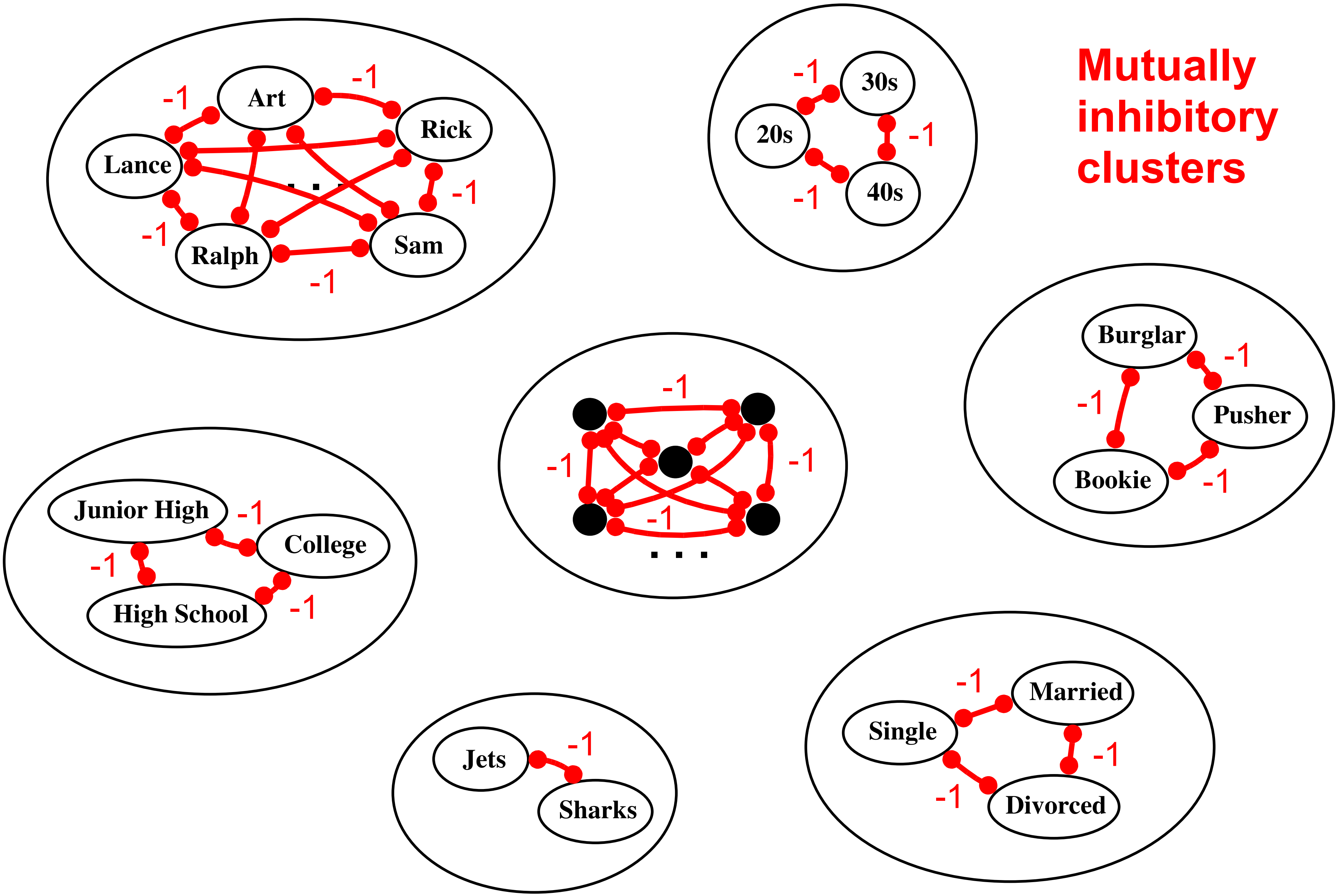


# The Jets and the Sharks

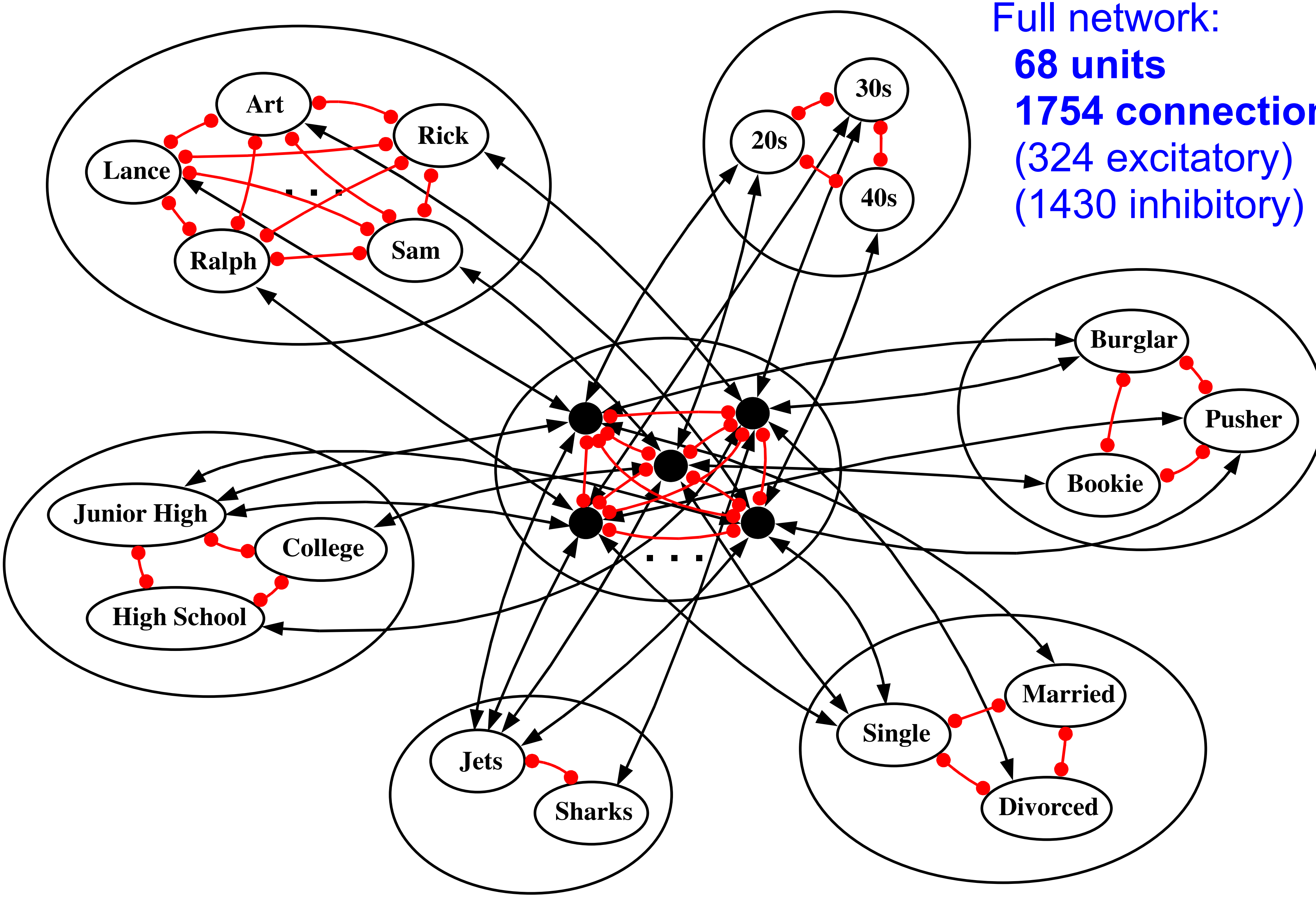


# The Jets and the Sharks

**Mutually inhibitory clusters**

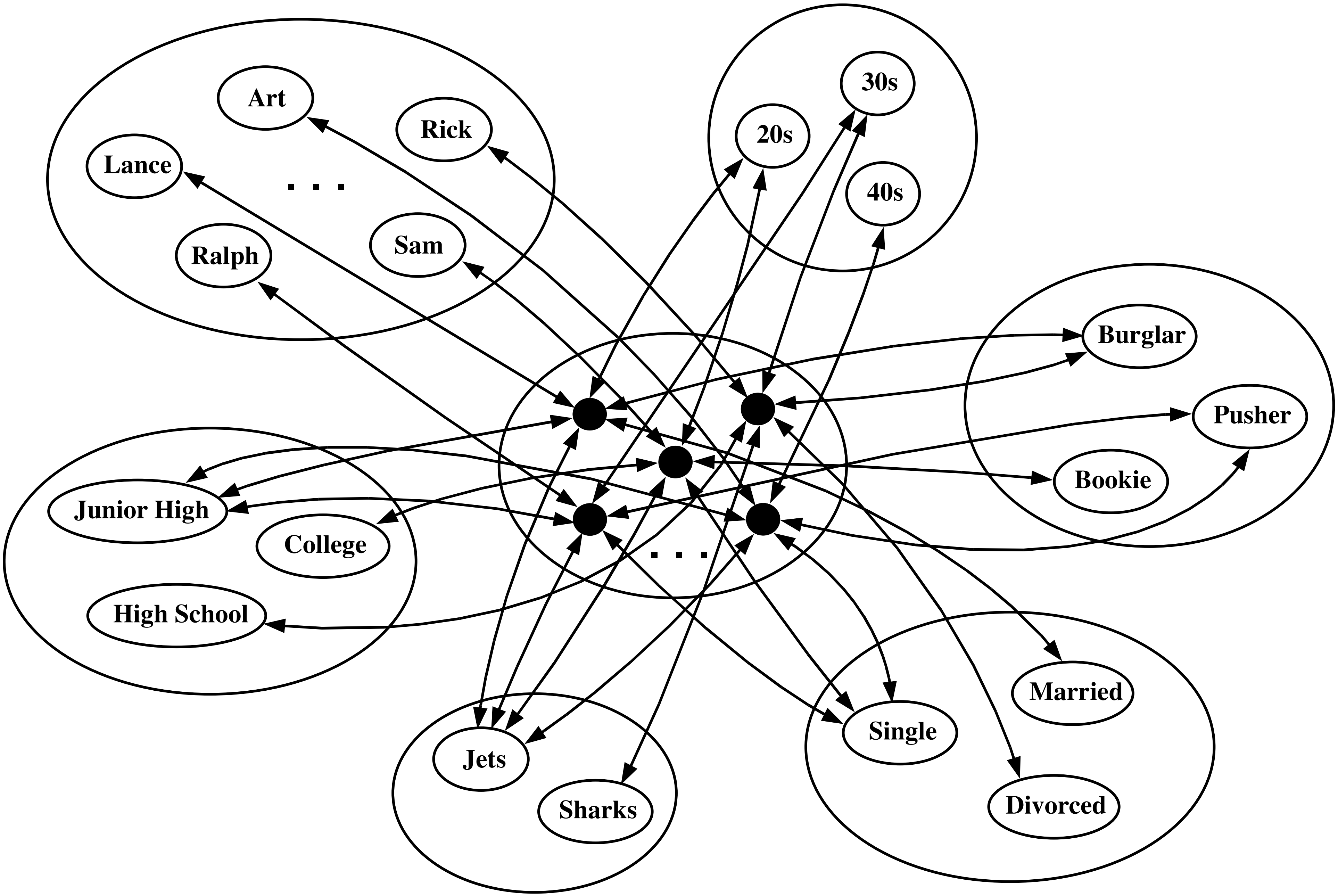


# The Jets and the Sharks

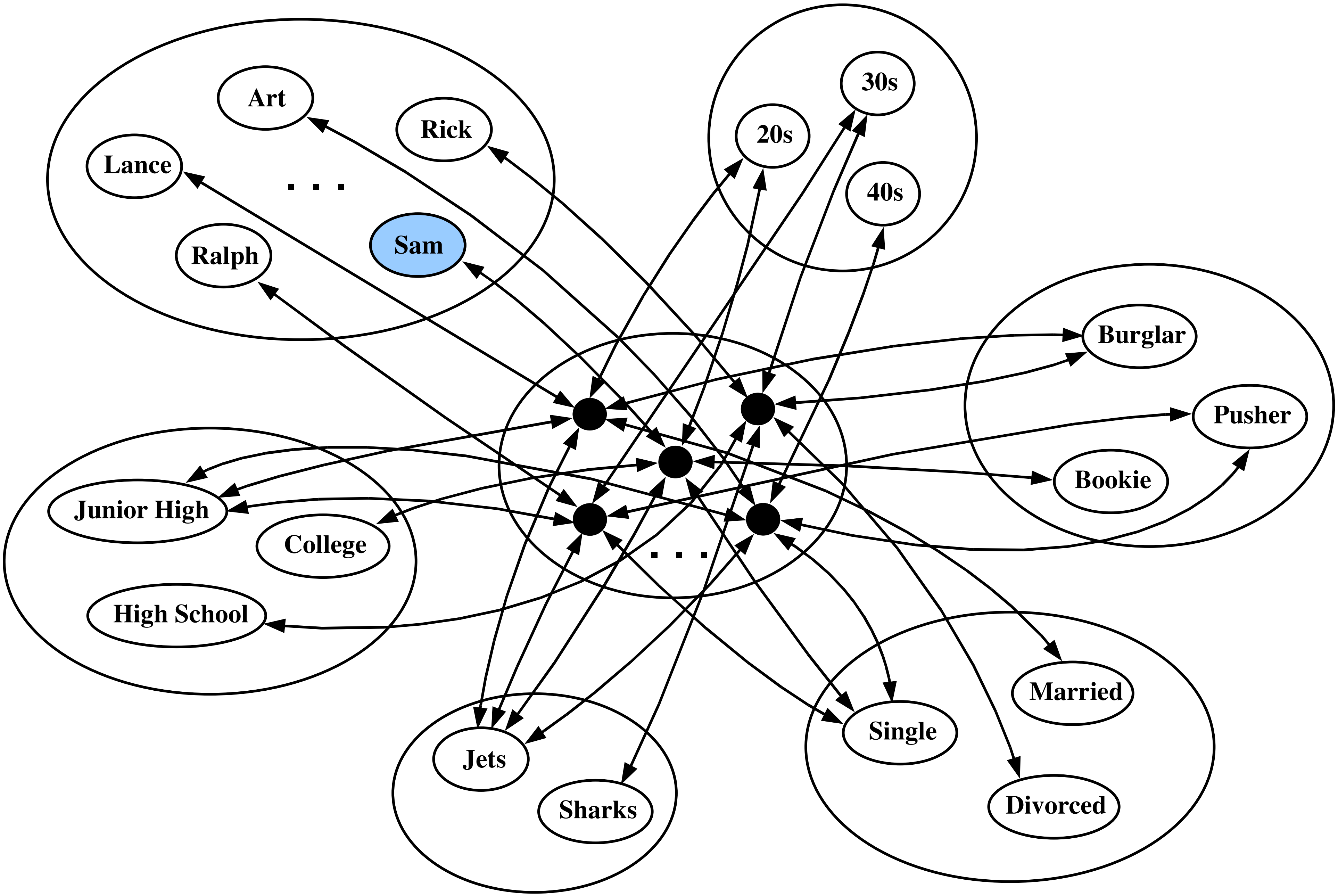



Full network:
**68 units**
**1754 connections**
(324 excitatory)
(1430 inhibitory)

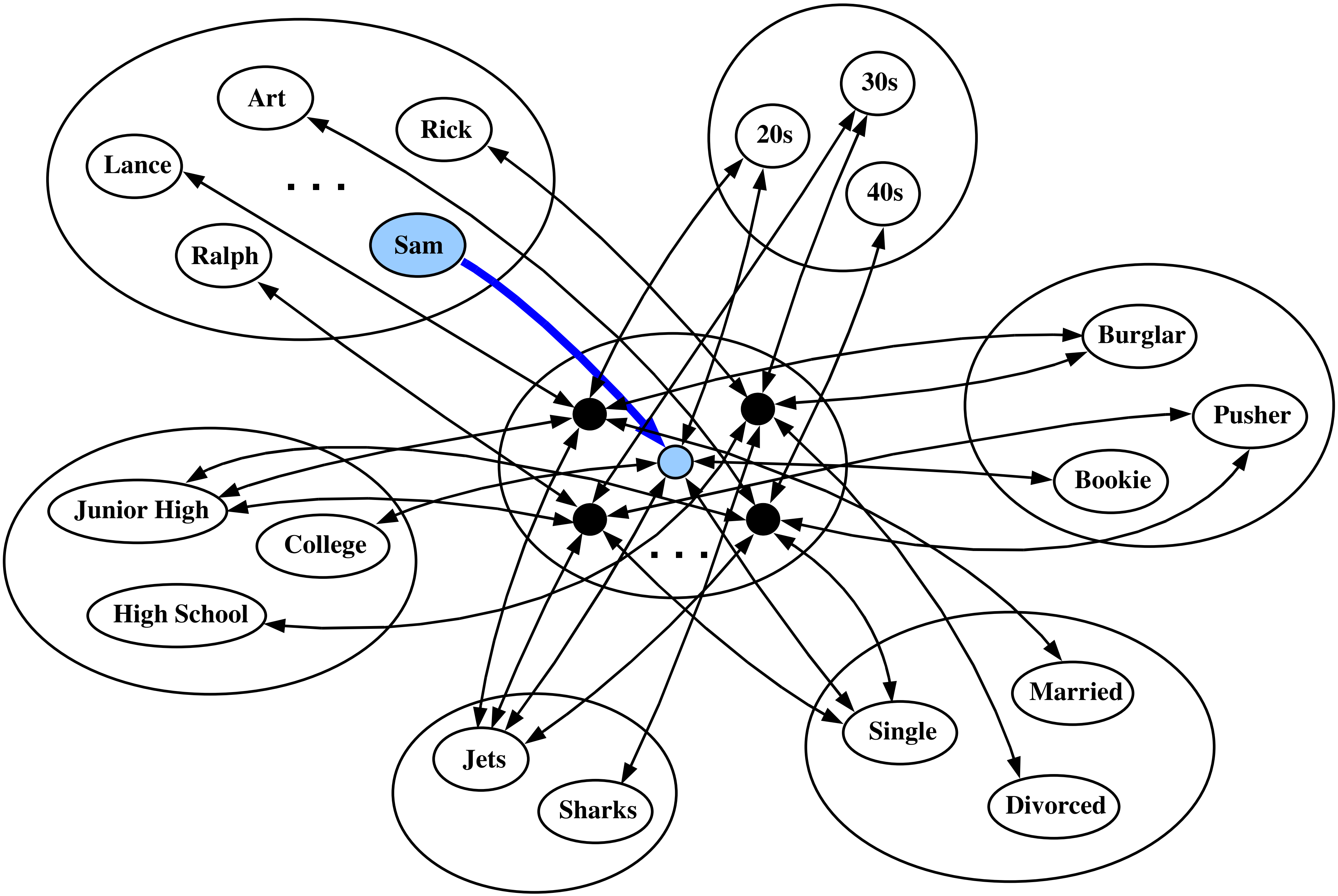
# Example 1: Retrieving Info About Sam



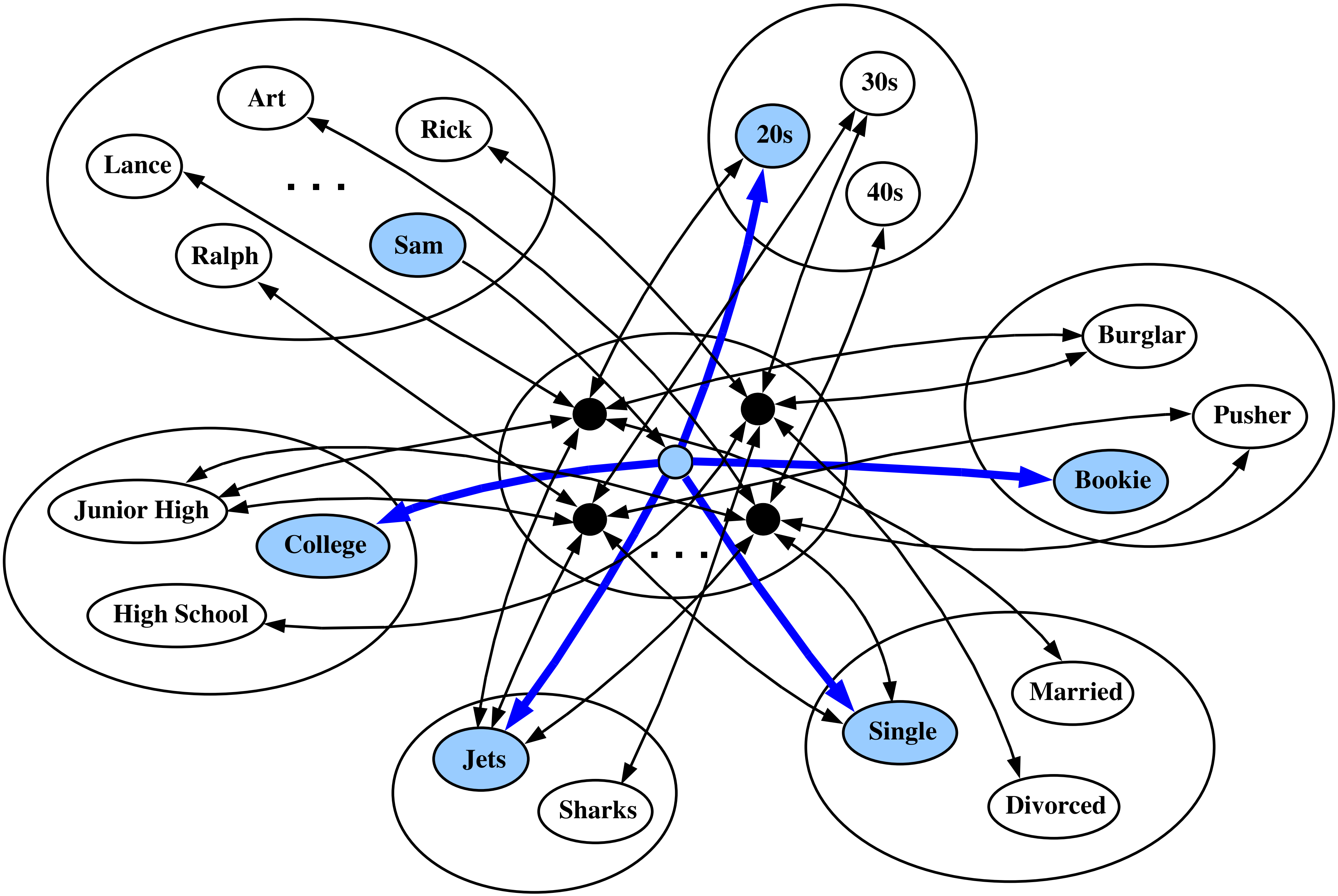
# Example 1: Retrieving Info About Sam



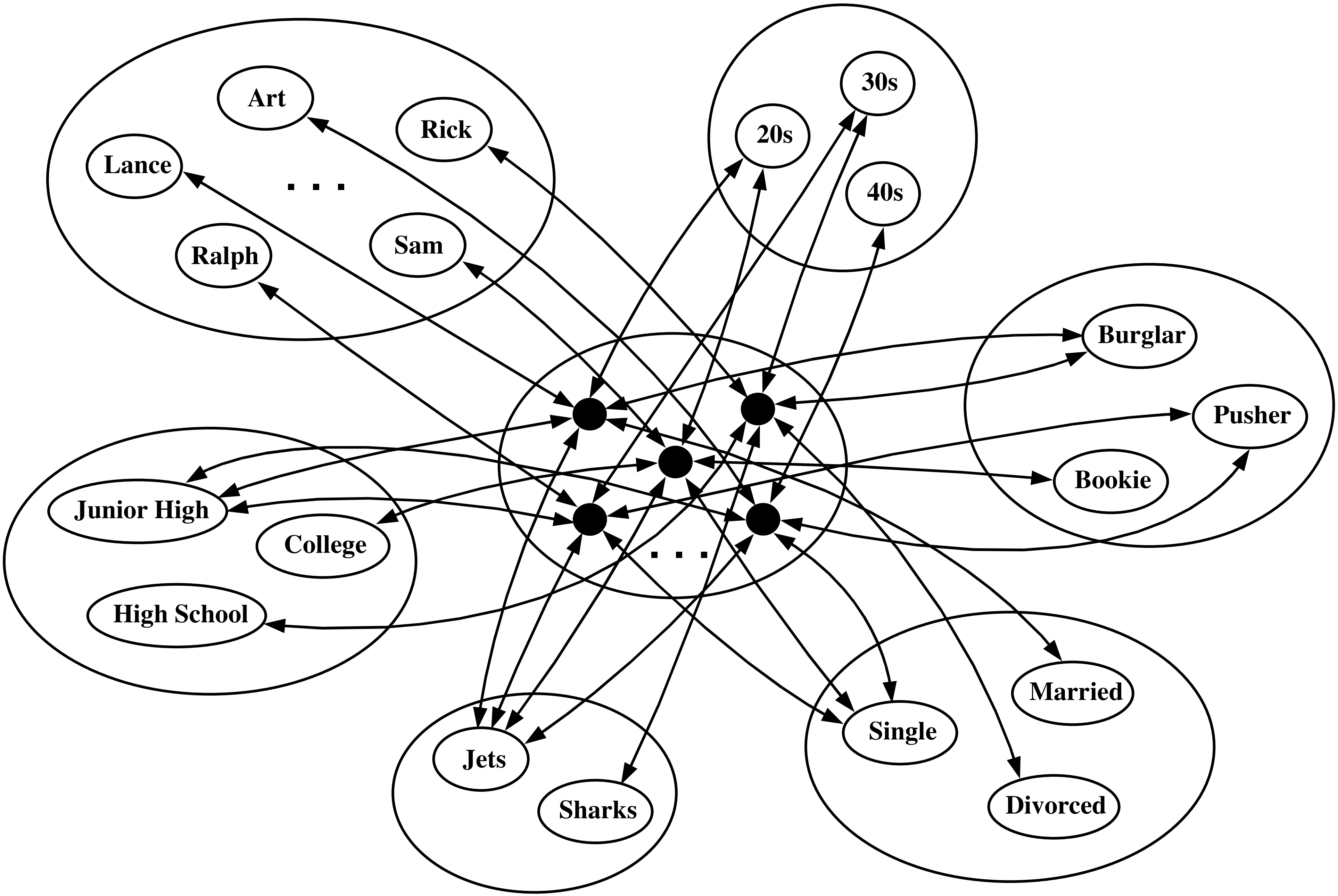
# Example 1: Retrieving Info About Sam



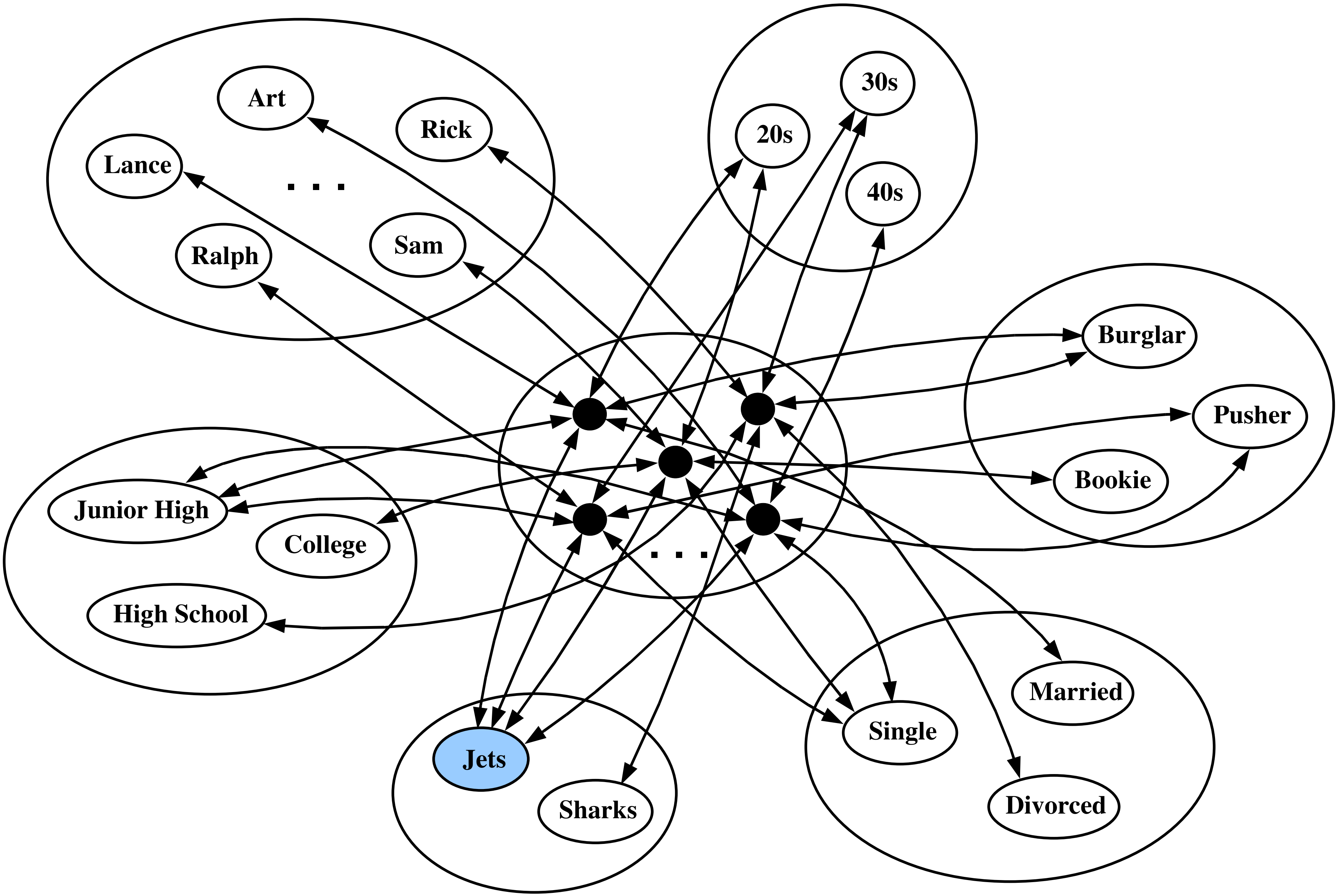
# Example 1: Retrieving Info About Sam



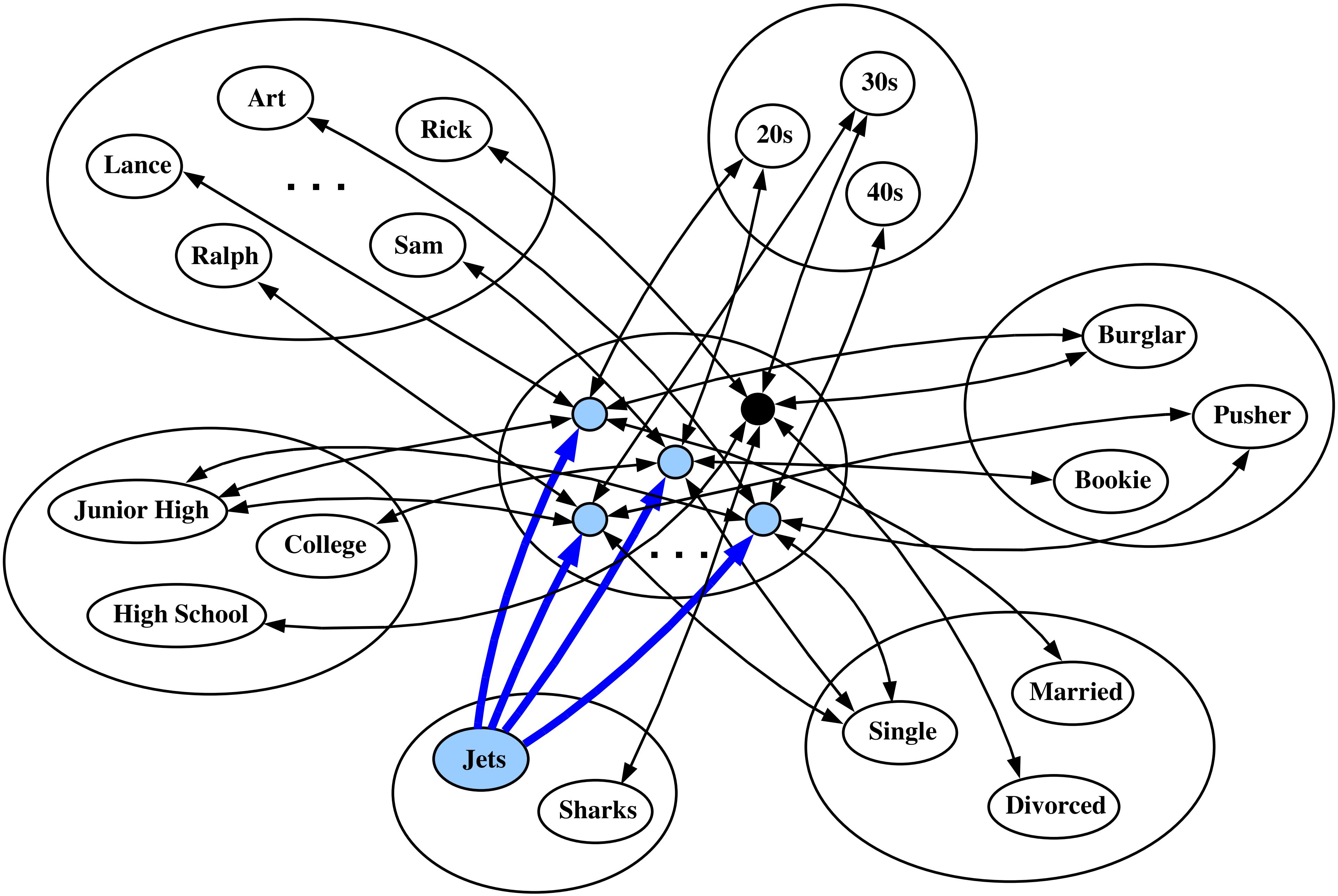
# Example 2: Retrieving Info About Jets



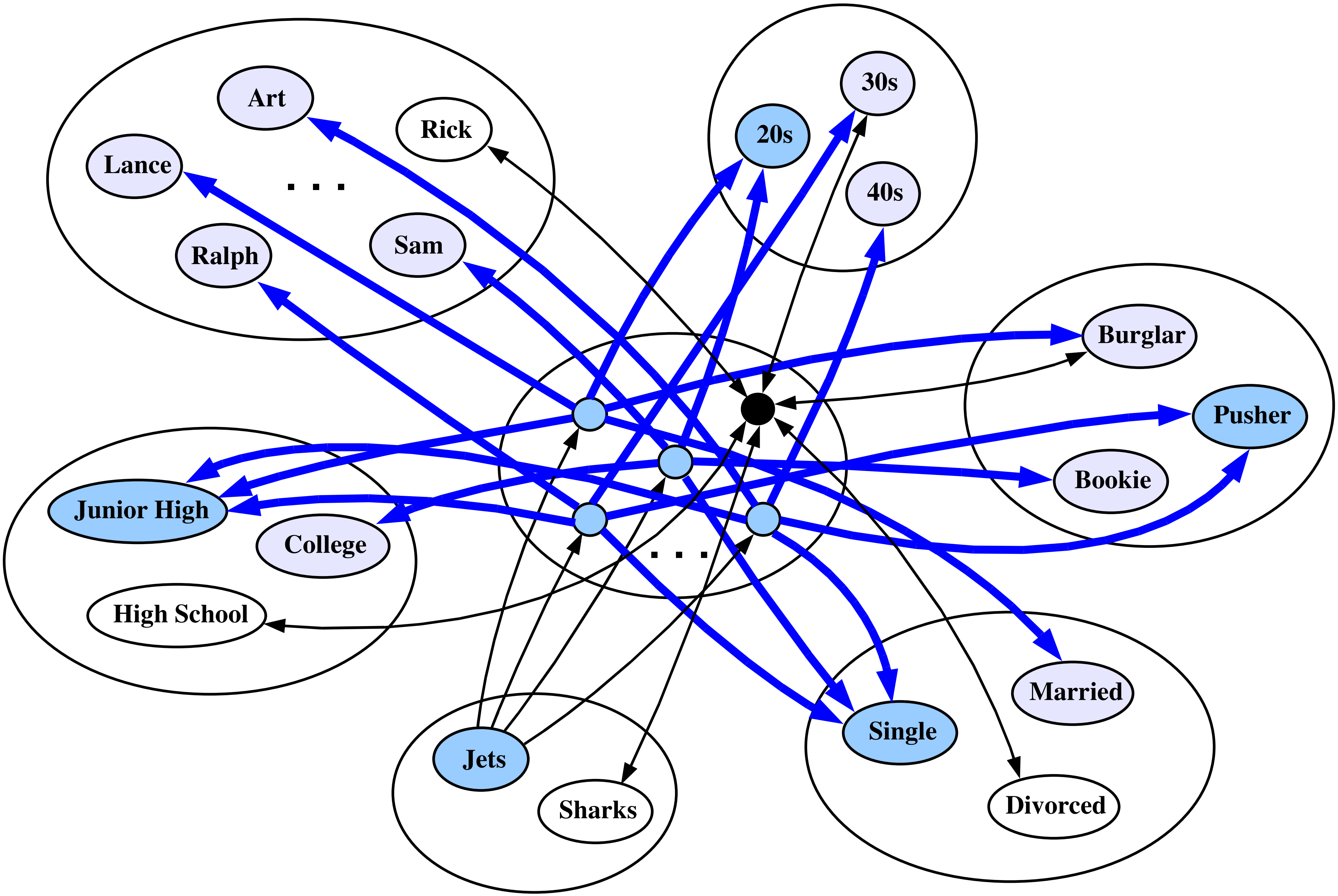
# Example 2: Retrieving Info About Jets



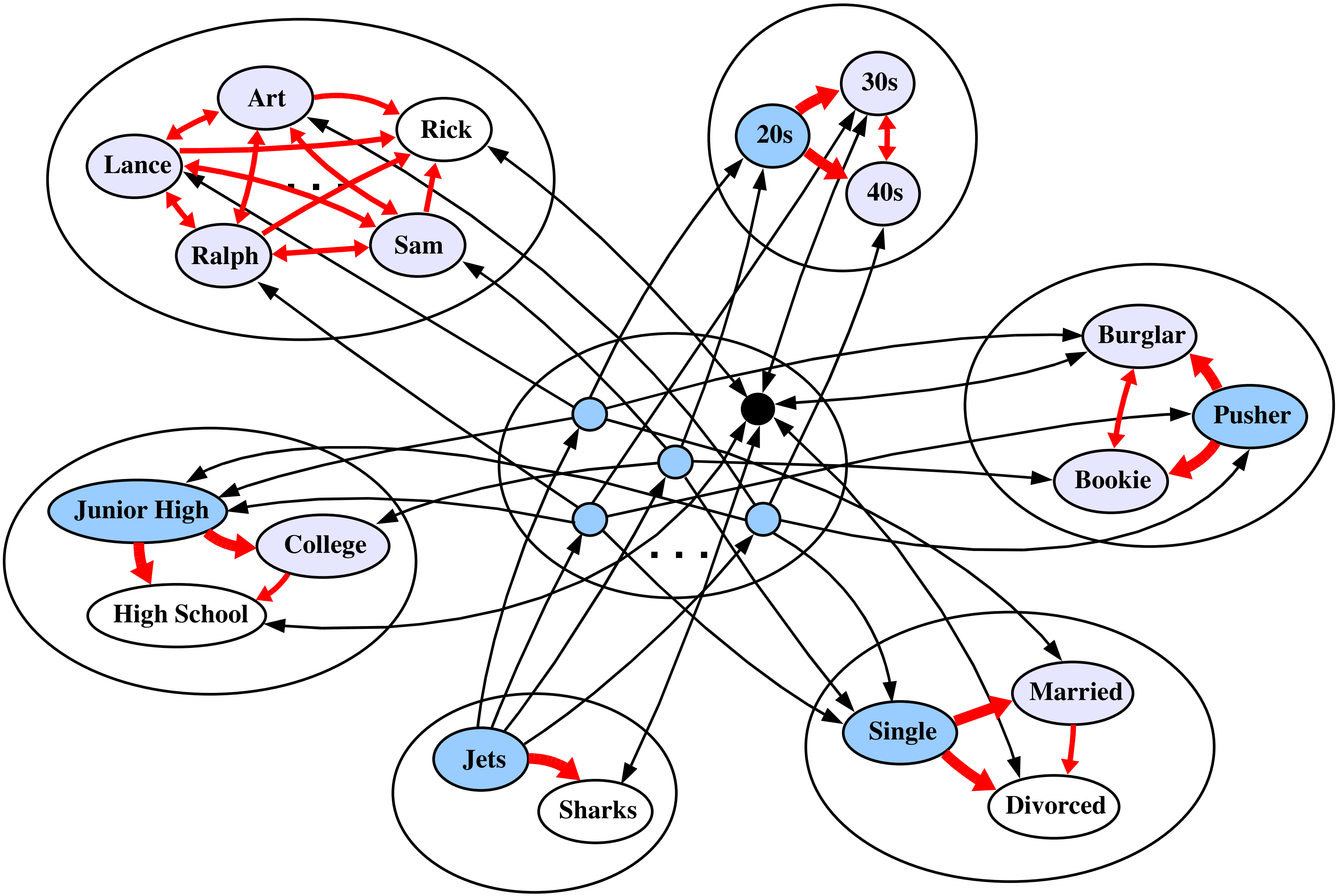
# Example 2: Retrieving Info About Jets



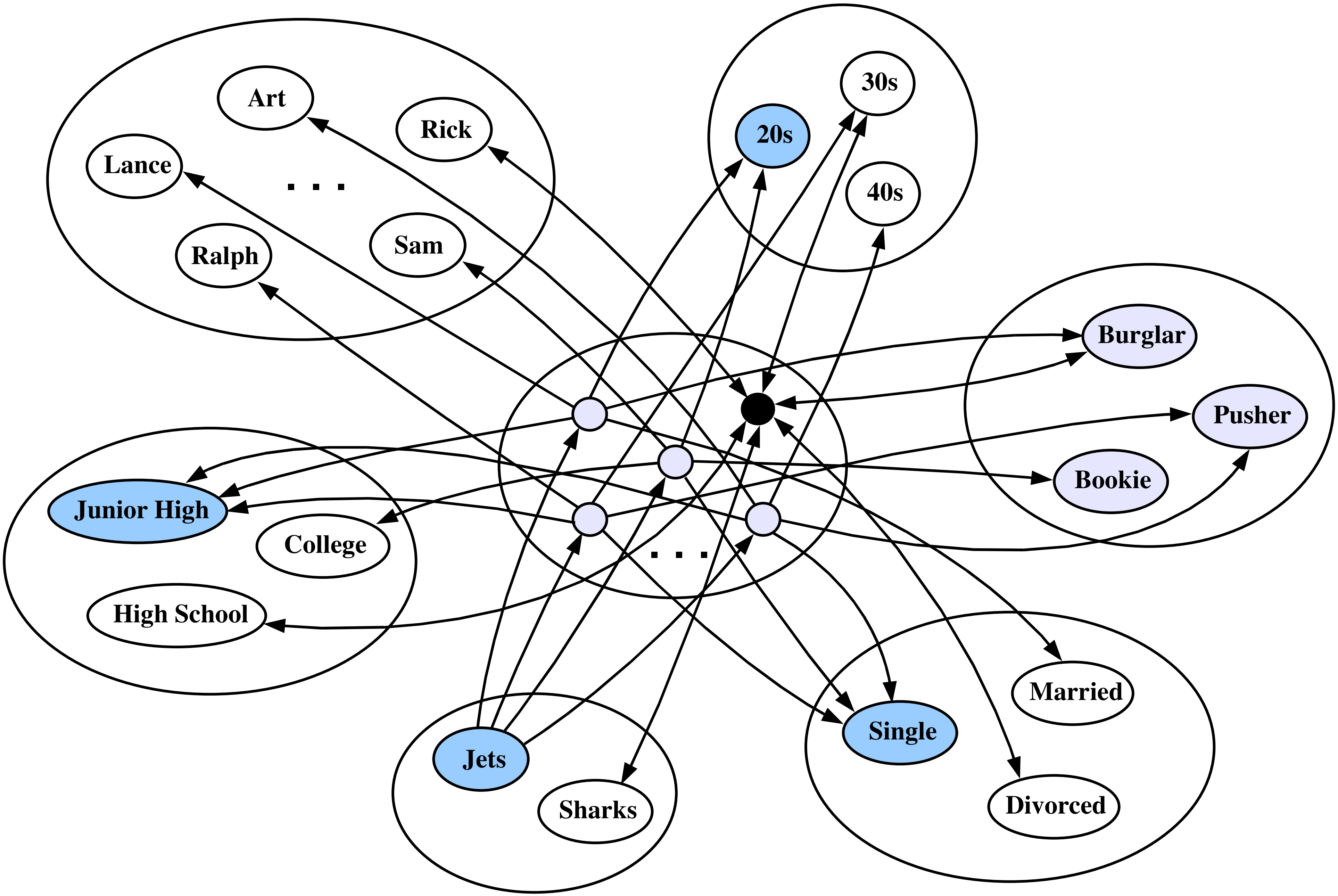
# Example 2: Retrieving Info About Jets



# Example 2: Retrieving Info About Jets



# Example 2: Retrieving Info About Jets



# Properties of the Memory

- **Graceful degradation**
  - damaging a few connections or units will degrade the memory's performance, but not in a catastrophic way
- **Resistance to noise**
  - probing the memory with partially incorrect information will usually still produce meaningful output
- **Spontaneous generalization**
  - the memory is able to retrieve prototypical patterns based on common similarities shared among several memories
- **Pattern completion**
  - the memory can fill in missing properties of individuals based on what it knows about other, similar instances

# Jets and Sharks Demo